Security Mechanisms for Distributed Communication Systems

Johann van der Merwe

B.Sc. Engineering (Electronic) and M.Sc. Engineering (Electronic),

University of Natal and University of KwaZulu-Natal, South Africa

Submitted in fulfillment of the academic requirements
for the degree of PhD in Electronic Engineering
in the School of Electrical, Electronic and Computer Engineering
at the University of KwaZulu-Natal, South Africa

June 27, 2010

Opgedra aan my vrou Roxane, vir al haar liefde en ondersteuning.

This document was created in LaTeX 2ε

Acknowledgments

I would like to express my deep and sincere gratitude to my supervisor, Professor Dawoud S. Dawoud. He has directed and encouraged me to complete the PhD degree over a period of four years and prior to this the MSc degree over two years. I know that during this period of time I have brought him joy, but also a fair share of frustration. Nevertheless, without fail he supported me to overcome the significant challenges of completing the PhD degree part-time. I will always hold Professor Dawoud in high regard and wish him only the best for the future.

I am grateful to ARMSCOR, the Armaments Corporation of South Africa, who financed part of this research project.

Ek wil ook graag baie dankie sê aan my vriende en familie, my ma, Georgina, suster, Jana en broer, Richart vir hulle liefde en ondersteuning.

Laastens sê ek dankie aan my vrou, Roxane vir haar eindlose geduld, aanmoediging and liefde. Haar vrienskap lê my na aan die hart. Geen woorde kan beskryf hoe lief ek vir haar is nie.

Johann van der Merwe
June 27, 2010

Preface

The research work presented in this thesis was performed by Johann van der Merwe, under the supervision of Prof. Dawoud S. Dawoud, in the School of Electrical, Electronic and Computer Engineering, University of KwaZulu-Natal. This work was supported by ARMSCOR, the Armaments Corporation of South Africa.

I, Johann van der Merwe, declare that:

(i) The research reported in this thesis, except where otherwise indicated, is my original work.

(ii) This thesis has not been submitted for any degree or examination at any other university.

(iii) This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.

(iv) This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

  a) their words have been re-written but the general information attributed to them has been referenced; and

  b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced.

(v) Where I have reproduced a publication of which I am an author, co-author or editor, I have indicated in detail which part of the publication was actually written by myself alone and have fully referenced such publications.

(vi) This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

Signed: _____

Name: Johann van der Merwe

Date: June 27, 2010

As the candidate's Supervisor I have approved this thesis for submission.

Signed: _____

Name: Prof. Dawoud S. Dawoud

Date: June 27, 2010

Abstract

In this thesis we study security mechanisms related to distributed communication systems within the context of peer-to-peer and group communication. These mechanisms include authority-based peer-to-peer key management, group key management, distributed-key management and threshold-multisignature schemes. The thesis is comprised of four parts.

The first part of the thesis proposes a peer-to-peer key management scheme for authority-based mobile ad hoc networks. The key management scheme bootstraps and maintains the security associations in the network, that is, it creates, distributes and revokes keying material as needed by the networking services. The proposed key management scheme breaks the routing-security interdependency cycle and exploits the unpredictable and dynamic network topology to the advantage of security.

The second part of the thesis presents a group key management scheme for dynamic peer groups that is suitable for ad hoc networks. The group key management scheme exploits the dynamic group membership and network topology to assist with the bootstrapping of security associations for the group communication system protocols. These protocols include unicast routing, group membership service, multicasting, group key agreement and data sharing. We also show how to bootstrap the group communication system by proposing a progressively robust, primary-partition group membership service. The membership service exploits the inherent capability of the group communication system to mitigate the impact of frequent membership changes and routing failures.

The third part of the thesis considers distributed-key (secret sharing) management mechanisms for generic, distributed communication systems. Specific attention is given to secret sharing in a setting without any form of online authority. The proposed Distributed-Key Management Infrastructure (DKMI) gives group members the capability to share, update and redistribute a secret in support of a threshold cryptosystem.

The fourth part of the thesis presents a threshold-multisignature scheme that allows group signatures to be generated in a collaborative fashion. The proposed scheme guarantees the signature verifier that at least a defined threshold of group members participated in the generation of the group-oriented signature and that the identities of the signers are traceable. The characteristics of secure and robust threshold-multisignature schemes are defined and it is shown that the proposed scheme satisfies these properties.

Finally, the thesis analyzes the proposed schemes from a performance and security perspective in widely acceptable system and adversary models.

Contents

Acknowledgments ii

Preface iii

Abstract vi

Contents viii

List of Figures xv

1 Introduction 1

1.1 Scope of Research 3

1.2 Publications 5

1.3 Outline of Thesis 6

I Peer-to-Peer Key Management in Distributed Communication Systems 8

2 Dynamic Network Topology Advances Security in Mobile Ad Hoc Networks 9

2.1 Introduction 9

2.2 Related Work 13

2.3 Proposed Offline Authority-Based Public Key Establishment Scheme 14

2.4 Proposed Authority-Based Public Key Management Scheme 16

2.4.1 System and Adversary Model 16

2.4.2 Offline Initialization Phase of AuthBasedPKM 17
2.4.3 Online Post-initialization Phase of AuthBasedPKM 18
2.4.3.1 Certificate distribution 18
2.4.3.2 Certificate authentication 21
2.4.3.3 Certificate revocation 21
2.4.3.4 Certificate renewal 23
2.5 Discussion on the Security and Features of AuthBasedPKM 25
2.5.1 On the Security of AuthBasedPKM 26
2.5.1.1 On the security of the proposed authority-based public key establishment (APKE) scheme 28
2.5.1.2 On the security of subordinate public keys and hash based identifiers 31
2.5.1.3 On the security of CertRelay 32
2.5.2 On the Performance of AuthBasedPKM 35
2.5.2.1 Efficiency of AuthBasedPKM initialization phase 35
2.5.2.2 Efficiency of AuthBasedPKM post-initialization phase 35
2.5.2.3 Simulations 35
2.6 Conclusion 41

II Group Key Management in Distributed Communication Systems 45

3 A Survey on Group Key Management for Mobile Ad Hoc Networks 46

3.1 Introduction 46
3.1.1 Organization of Chapter 51
3.2 Contributory Key Agreement: Serial Topology 52
3.2.1 CLIQUES Initial Key Agreement: IKA.1 53
3.2.2 CLIQUES Initial Key Agreement: IKA.2 54
3.2.3 CLIQUES Auxiliary Key Agreement: Membership Addition 55
3.2.4 CLIQUES Auxiliary Key Agreement: Merge Protocol - GDH IKA.3 56
3.2.5 CLIQUES Auxiliary Key Agreement: Partition Protocol 57

3.2.6 Discussion of the CLIQUES Protocol Suite 58
3.3 Contributory Key Agreement: Circular Topology 59
3.3.1 Ingemarsson Initial Key Agreement: *ING* 59
3.3.2 Discussion of the Ingemarsson Protocol 60
3.4 Contributory Key Agreement: Hypercube Topology 61
3.4.1 Hypercube Initial Key Agreement 62
3.4.2 Discussion of the Hypercube Protocol 63
3.5 Contributory Key Agreement: Octopus Topology 63
3.5.1 2^d-Octopus Protocol 64
3.5.2 Discussion of the Octopus - and 2^d-Octopus Protocols 65
3.6 Contributory Key Agreement: Tree-based Topology 66
3.6.1 *TGDH* Auxiliary Key Agreement: Membership Addition 68
3.6.2 *TGDH* Auxiliary Key Agreement: Membership Exclusion 70
3.6.3 Discussion of the *TGDH* Protocol Suite 71
3.7 Contributory Key Agreement: Arbitrary Topology 72
3.7.1 AT-GDH Initial Key Agreement 73
3.7.2 Discussion of the AT-GDH Initial Key Agreement Protocol 75
3.8 Contributory Key Agreement: Star Topology 76
3.8.1 GKA Initial Key Agreement Protocol 77
3.8.2 GKA Auxiliary Key Agreement Protocol: Join/Merge 78
3.8.3 Discussion of the GKA Protocol Suite 80
3.9 Contributory Key Agreement: Topology Independent Schemes 81
3.9.1 *BD* Initial Key Agreement 81
3.9.2 *BD* Auxiliary Key Agreement: Membership Addition 82
3.9.3 Discussion of the *BD* Protocol Suite 82
3.10 Key Pre-Distribution Schemes 84
3.10.1 Distributed Key Selection (DKS) 84
3.10.2 Secure Shared-key Discovery (SSD) 86

3.10.3 Key Exclusion Property Testing (KEPT) . . . 87
3.10.4 Computing the Group Key . . . 87
3.10.5 Discussion of the Distributed Key Pre-Distribution Scheme . . . 87
3.11 Decentralized Key Distribution . . . 89
3.11.1 CKD Auxiliary Key Transport: Mass Join . . . 90
3.11.2 CKD Auxiliary Key Transport: Mass Leave . . . 91
3.11.3 Discussion of the CKD Protocol Suite . . . 91
3.12 Performance Analysis . . . 92
3.12.1 Efficiency of IKA operations . . . 92
3.12.2 Efficiency of AKA operations . . . 93
3.13 Conclusions . . . 94

4 Bootstrapping Group Communication and Security in Mobile Ad Hoc Networks 101

4.1 Introduction . . . 101
4.1.1 Problem Statement . . . 102
4.1.1.1 Security Requirements for Group Communication . . . 102
4.1.1.2 Bootstrapping Security for Unicast Routing . . . 103
4.1.1.3 Group Membership Service for Bootstrapping Group Communication Systems . . . 103
4.1.1.4 Bootstrapping Security for Multicasting . . . 104
4.1.1.5 Bootstrapping Security for Contributory Group Key Agreement . . . 104
4.1.1.6 Bootstrapping Security for Dynamic Peer Groups . . . 105
4.1.2 Our Contribution . . . 105
4.1.3 Organization of Chapter . . . 106
4.2 Related Work . . . 106
4.3 AdHocGKM: Group Key Management in Ad Hoc Networks . . . 107
4.3.1 System and Adversary Models . . . 108
4.3.1.1 System Model . . . 108
4.3.1.2 Adversary Model . . . 109

4.3.2 Offline Initialization Phase of AdHocGKM 109
4.3.3 Online Post-initialization Phase of AdHocGKM 111
4.3.3.1 AdHocGKM Primary-partition Membership Service: Bootstrapping the GCS 111
4.3.3.2 Bootstrapping the Security of the Unicast Routing Protocol . . . 113
4.3.3.3 Bootstrapping the Security of the Membership Service, Multicasting and Data Sharing Protocols 114
4.3.3.4 Bootstrapping the Security of the Group Key Agreement Protocol 114
4.4 Discussion on the Features and Security of AdHocGKM 118
4.4.1 On the Features of AdHocGKM 118
4.4.1.1 Progressively Robust Key Distribution 119
4.4.1.2 Progressively Robust Group Membership Service 119
4.4.1.3 Progressively Robust Group Key Agreement 121
4.4.2 On the Security of AdHocGKM 121
4.4.2.1 On the Security of CertRelay 121
4.4.2.2 On the Security of the AdHocGKM group membership service . . 122
4.4.2.3 On the Security of the BD group key agreement protocol 122
4.4.3 On the Performance of AdHocGKM 124
4.5 Conclusions 125

III Distributed-Key Management in Distributed Communication Systems 128

5 Distributed-Key Management in Distributed Communication Systems 129

5.1 Introduction 129
5.2 Proposed Distributed-key Management Infrastructure 131
5.2.1 System model 131
5.2.2 Modified Zhang *et al.* Publicly Verifiable Distributed-Key Generation Protocol 132
5.3 Proposed Publicly Verifiable Distributed-Key Redistribution/Update Protocol . . . 134

5.4 Discussion on the Security and Features of the Proposed Distributed-key Management Infrastructure . . . 137
5.4.1 Initial key sharing security . . . 137
5.4.2 Proactive security and secret redistribution . . . 138
5.4.3 Efficiency analysis . . . 140
5.5 Conclusion . . . 141

IV Threshold-Multisignatures for Distributed Communication Systems 142

6 A Fully Distributed Proactively Secure Threshold-Multisignature Scheme 143

6.1 Introduction . . . 143
6.2 Proposed Threshold-Multisignature Scheme . . . 145
6.2.1 System model . . . 145
6.2.2 Initial publicly verifiable distributed-key generation . . . 145
6.2.3 Individual signature generation . . . 145
6.2.4 Individual signature verification . . . 146
6.2.5 Threshold-multisignature generation . . . 146
6.2.6 Threshold-multisignature verification and individual signer identification . . . 147
6.3 Proposed Publicly Verifiable Distributed-Key Redistribution/Update Protocol . . . 147
6.4 Discussion on the Security and Features of the Proposed Threshold-Multisignature Scheme . . . 148
6.4.1 Correctness and threshold property . . . 148
6.4.2 Traceability of signers . . . 149
6.4.3 Coalition-resistance and break-resistance . . . 150
6.4.4 Attacks on threshold-multisignature schemes . . . 151
6.4.4.1 Collusion attack . . . 151
6.4.4.2 Universal forgery attack . . . 152
6.4.4.3 Rushing attack . . . 152
6.4.4.4 Conspiracy attack . . . 154

6.4.5 Symmetric relationships - eliminating the combiner 155
6.4.6 Proactive security and secret redistribution 155
6.4.7 Efficiency analysis 155
6.4.7.1 Group public key length 155
6.4.7.2 Group-oriented signature size 156
6.4.7.3 Communication cost of signature generation and verification . . . 157
6.4.7.4 Computational cost of signature generation and verification 157
6.4.7.5 Efficiency of initial key distribution and key redistribution/update protocols 159
6.5 Conclusion 161

7 Conclusions and Future Direction **162**

7.1 Summary of Contributions 165
7.2 Direction for Future Research 167

List of Figures

2.1 CertRelay certificate distribution main procedure 18

2.2 CertRelay's CBR packet delivery ratio % vs. load in pkt/sec for 0.1m/sec, 5m/sec and 20m/sec mobility 39

2.3 CertRelay's CBR packet end-to-end delay vs. load in pkt/sec for 0.1m/sec, 5m/sec and 20m/sec mobility 39

2.4 Certificates sent by CertRelay vs. time in pkt/sec for 0.1m/sec, 5m/sec and 20m/sec mobility 40

2.5 CertRelay's certificate delivery ratio % vs. load in pkt/sec for 0.1m/sec, 5m/sec and 20m/sec mobility 40

2.6 CBR packet delivery ratio % (with and without public keys in AODV headers) vs. load in pkt/sec for 5m/sec mobility 41

2.7 CBR packet delivery ratio % (with and without public keys in AODV headers) vs. load in pkt/sec for 20m/sec mobility 42

2.8 CBR packet end-to-end delay (with and without public keys in AODV headers) vs. load in pkt/sec for 5m/sec mobility 42

2.9 CBR packet end-to-end delay (with and without public keys in AODV headers) vs. load in pkt/sec for 20m/sec mobility 43

3.1 Dimensions of group key management. 47

3.2 Common auxiliary key agreement operations [1]. 48

3.3 Cliques IKA.1 Algorithm - Case of a group with 4 members. 54

3.4 Round 1 of 2^2 Hypercube protocol 61

3.5 Round 2 of 2^2 Hypercube protocol 61

3.6 Network topology of Octopus protocol 63

3.7 *TGDH* key tree model example 66

3.8 *TGDH* update: join example 69

3.9 *TGDH* update: leave example 71

3.10 Constructing a spanning tree 72

3.11 Phase 1, Round -1: For all nodes with $C_x = 0$ (i.e. has no children) 74

3.12 Phase 1, Round-i: Only nodes with children 74

3.13 Phase 2, Round -1: For all nodes with $C_x = 0$ (i.e. has no children) 75

3.14 Phase 2, Round-i: Only nodes with children 75

3.15 GKA IKA Key Agreement Protocol with $n = 7$ and U_0 as group leader 79

3.16 DKS based on probabilistic CFF construction 85

4.1 Protocol structure of AdHocGKM integrated with PILOT [2] 108

Chapter 1

Introduction

Advances in mobile computing and wireless communication technology have led to the development of new and innovative applications based on distributed communication services. The main characteristic of distributed communication systems is the lack of centralized infrastructure or control.

The available literature contains numerous examples of distributed communication systems. Some of the most prominent research areas include mobile ad hoc networks [3] [4] [5] and vehicular communications [6] [7]. On a network level, these networks may be fully or partially supported by infrastructure, while *ad hoc networks* rely on no fixed infrastructure.

On an application level, communication mechanisms may depend on limited centralized control. Peer-to-peer (P2P) systems [8] and dynamic peer groups (DPGs) [1] [9] are examples of such communication mechanisms.

Many of the intended uses of distributed communication services require them to be appropriately secured. For example, without integrating security into vehicular networks, these systems can be compromised, jeopardizing the safety of the drivers and passengers. Another example is a conferencing system that relies on the security mechanisms provided by the underlying group communication system [10]. Compromise of the security mechanisms may, for example, result in theft of confidential information shared between the group members or the group members may even be held accountable for digitally signing a fraudulent document.

The nature of distributed wireless communication systems, depending on the characteristics of

the network and/or the application, poses many challenges to the design of appropriate security mechanisms:

- On a network level the major challenges are as a result of the error prone, inherently insecure connectivity offered by wireless communication technologies, user mobility and limited (or complete lack of) pre-existing infrastructure [3] [4] [5]. In ad hoc networks the users themselves may set up the self-organized network. Without supporting infrastructure the major problem lies with bootstrapping the security mechanisms during network formation.
- On an application level these challenges are mainly as a result of dynamic membership and the need for self-organized communication with limited (or nonexistence of) centralized control.

In this thesis the focus is on a selection of the fundamental security mechanisms required to protect distributed communication systems. These mechanisms include key management in a peer-to-peer and group setting as well as group-oriented digital signatures.

Key management can be defined as a set of techniques and procedures supporting the establishment and maintenance of keying material between authorized parties [11]. The cryptographic security mechanisms enabled by key management are useful for achieving many security objectives such as authentication, confidential communications and message integrity. Key management protocols are required to support peer-to-peer applications and group communications.

Peer-to-peer key management, as defined in [5], is considered for applications and/or networks that depend on an authority to assist with the initialization of system users (network devices) and the generation of initial keying material [11]. The offline authority essentially controls access to the network or limits the use of the application to authorized users only. Examples of such networks include ad hoc networks formed by users that serve a common purpose such as large scale emergency response operations and military missions. Vehicular networks will also fall under this classification. Peer-to-Peer key management for *fully self-organized* networks, as defined in [4], is not considered in this thesis. Our preliminary work [12] [13] addresses key management in self-organized ad hoc networks. In fully self-organized networks users form the network by themselves without assistance from an authority. Applications for such networks are community based (social) applications suitable for spontaneous user interactions.

Group communication relies on key management schemes to share and distribute keys in support of the basic security mechanisms (for example encryption of shared data). As these groups may be formed without prior arrangement, an online authority will not always be available to share

or distribute keying material. Examples of such group communications may include applications where a group of individuals want to privately share a conversation or documents. In this thesis group key management is considered for both shared and distributed-keys as defined in [1] and [14] respectively.

Digital signatures provide a means to ensure the integrity of data and to bind entities to actions [11]. Although digital signatures for single entities are a fully mature area, group signatures are still posing challenges. This thesis considers group signature generation and verification, as defined in [15], for dynamic settings where a threshold of group members are required to sign messages, each using their share in the distributed group key.

1.1 Scope of Research

The scope of this work is security mechanisms for distributed, authority-based communication systems which include the following areas:

- Peer-to-peer key management for authority-based ad hoc networks with key distribution as a subset;
- Group key management and group membership services for dynamic peer groups;
- Distributed-key management for dynamic peer groups; and
- Group signature schemes for dynamic peer groups.

Our main aim is to design mechanisms that exploit the unpredictable and dynamic network topology of ad hoc networks and the inherent capability of network services to recover from failures. We define the capability of a protocol to improve its performance in an increasingly hostile environment as *progressive robustness*.

As an introduction, we discuss authority based systems followed by each of the research areas listed above.

Authority-based Systems [5] [11]: Key management in traditional wired networks is normally the responsibility of a centralized authority. If the authority is off-line it can issue nodes with shared keys or certificates. If the authority is also present during on-line operations it can assist with establishing keying relationships between networking participants during network formation and

node encounters. In this thesis the goal is to explore peer-to-peer or pairwise key management in distributed communication systems where security relies only on an off-line authority with limited functionality. We refer to this type of network as authority-based systems [5] [16]. We assume that the off-line authority may violate the trust relationship it has with the network participants or may be compromised by a malicious party during the lifetime of the network. This means the off-line authority should not have access to the private keys of users. The main applications of this type of key management schemes are self-organized ad hoc networks in which the networks emerge dynamically over time as users join the network. The off-line authority may therefore be involved in network initialization over an extended period of time, while the network is already in operation. There are many examples of such networks in literature for example peer-to-peer networking, vehicular networking, emergency response and military missions.

Group key management [1]: Extending pairwise key management to group key management is not a trivial task [1]. This is particularly true where security relies only on an off-line authority with limited functionality and without giving the authority access to the group key. This thesis explores group key management for dynamic peer groups, which are small groups with membership in the hundreds rather than thousands. The dynamic aspect of the group is derived from the fact that members may join and leave the group many times during the group's lifetime. We are particularly interested in the implementation of group key agreement in ad hoc networks and how to establish and maintain the group membership.

Distributed-key management [14]: Distributed-key management schemes are geared towards establishing a distributed-key between group members. A shared (common) key is useful for encrypting communications between the group members, but does not allow the group to digitally sign a message collectively. For this purpose a distributed-key is required. This key should obviously not be the same as that of the shared key used for encryption and each member should have an equal stake in the distributed-key. This thesis investigates how to set up and maintain such a distributed-key in a dynamic group setting.

Group signatures [15]: As the group signature is an assertion on behalf of the group, the signature scheme should enforce an equal contribution from members based on a predefined threshold of votes. Without any form of on-line authority, group members need to construct the group signature in a self-organized fashion. Part of our work is to explore the design of threshold-multisignature schemes that meet stronger accountability and robustness properties. This scheme should only allow a threshold of group members with a stake in the distributed-key to generate the group signature and allow any public entity to verify the group signature.

The application of distributed-key management and threshold-multisignature schemes include any group-oriented application that require basic security services that have strong integrity, accountability and non-repudiation requirements.

1.2 Publications

The following publications are directly as a result of the author's work in this PhD thesis:

[17] J. van der Merwe, and D. Dawoud Key Distribution in Mobile Ad Hoc Networks based on Message Relaying, in proc. Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS07), May, 22-25 2007.

[18] J. van der Merwe, D. Dawoud, and S. McDonald, A Fully Distributed Proactively Secure Threshold-Multisignature Scheme, IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 4, pp. 562-575, 2007.

[19] J. van der Merwe and D. Dawoud, Key Management for Dynamic Peer Groups in Mobile Ad Hoc Networks, in Mobile Peer-to-Peer Computing for Next Generation Distributed Environments: Advancing Conceptual and Algorithmic Applications, B.-C. Seet, Ed. IGI Global, 2009, pp. 241-281.

Submitted for review

[20] J. van der Merwe and D. Dawoud, Bootstrapping Group Communication and Security in Ad Hoc Networks, 2010, in submission.

[21] J. van der Merwe and D. Dawoud, Dynamic Network Topology Advances Security in Mobile Ad Hoc Networks, 2010, in submission.

The following publications are as a result of author's preliminary work during the period of the MSc degree and form a solid foundation for the research in this PhD thesis:

[16] J. van der Merwe, D. Dawoud, and S. McDonald, A Survey on Peer-to-Peer Key Management for Mobile Ad Hoc Networks, ACM Computing Surveys (CSUR), vol. 39, no. 1, pp. 132, 2007.

[**22**] J. van der Merwe, D. Dawoud, and S. McDonald, A Public Key Management Scheme and Threshold-Multisignature Scheme for Mobile Ad Hoc Networks, SAIEE Africa Research Journal (Transactions of SAIEE), vol. 97, no. 1, pp. 132, 2006.

[**12**] J. van der Merwe, D. Dawoud, and S. McDonald, Fully Self-Organized Peer-to-Peer Key Management for Mobile Ad Hoc Networks, in proc. ACM Workshop on Wireless Security (WiSe05), September, 2 2005.

[**13**] J. van der Merwe, Key Management in Mobile Ad Hoc Networks, M.Sc. in Engineering (Electronic), University of KwaZulu-Natal (UKZN), 2005.

[**23**] J. van der Merwe, D. Dawoud, and S. McDonald, Self-Organized Peer-to-Peer Key Management for Mobile Ad Hoc Networks, in proc. Southern African Telecommunication Networks and Applications Conference (SATNAC05), 2005.

[**24**] J. van der Merwe, D. Dawoud, and S. McDonald, A Survey on Peer-to-Peer Key Management for Military Type Mobile Ad Hoc Networks, in proc. Military Information and Communications Symposium of South Africa, 2005.

[**25**] J. van der Merwe, D. Dawoud, and S. McDonald, Group Key Management for Military Type Mobile Ad Hoc Networks, in proc. Military Information and Communications Symposium of South Africa, 2005.

[**26**] J. van der Merwe, D. Dawoud, and S. McDonald, Public Key Management for Military Type Mobile Ad Hoc Networks, in proc. Military Information and Communications Symposium of South Africa, 2005.

[**27**] J. van der Merwe, D. Dawoud, and S. McDonald, Trustworthy Key Management for Mobile Ad Hoc Networks, in proc. Southern African Telecommunication Networks and Applications Conference (SATNAC04), September, 6-8 2004.

1.3 Outline of Thesis

The thesis is organized as follows:

Part I: Peer-to-Peer Key Management in Distributed Communication Systems

Chapter 2 proposes a peer-to-peer key management scheme for authority-based ad hoc networks. The key management scheme bootstraps and maintains the security associations in the network,

that is, it creates, distributes and revokes keying material as needed by the networking services. The proposed key management scheme breaks the routing-security interdependency cycle and exploits the unpredictable and dynamic network topology to the advantage of security.

Part II: Group Key Management in Distributed Communication Systems

Chapter 4 of the thesis presents a group key management scheme for dynamic peer groups (DPGs). The scheme is founded on a comprehensive survey of existing schemes and their suitability for ad hoc networks given in Chapter 3. Our group key management scheme exploits the dynamic group membership and network topology to assist with the bootstrapping of security associations for the group communication system protocols. These protocols include unicast routing, group membership service, multicasting, group key agreement and data sharing. We also show how to bootstrap the group communication system by proposing a progressively robust, primary-partition group membership service. The membership service exploits the inherent capability of the group communication system to mitigate the impact of frequent group membership changes and routing failures.

Part III: Distributed-Key Management in Distributed Communication Systems

In Chapter 5, distributed-key (secret sharing) management mechanisms are proposed for generic, distributed communication systems. Specific attention is given to secret sharing in a setting without any form of online authority. The proposed Distributed-Key Management Infrastructure (DKMI) gives group members the capability to share, update and redistribute a secret in support of a threshold cryptosystem.

Part IV: Threshold-Multisignatures in Distributed Communication Systems

Chapter 6 presents a threshold-multisignature scheme that allows group signatures to be generated in a collaborative fashion. The proposed scheme guarantees the signature verifier that at least a defined threshold of group members participated in the generation of the group-oriented signature and that the identities of the signers are traceable. The characteristics of secure and robust threshold-multisignature schemes are defined and it is shown that the proposed scheme satisfies these properties.

All proposed schemes in Chapter 2 to 6 are analyzed from a security and performance perspective in widely acceptable system and adversary models.

Part I

Peer-to-Peer Key Management in Distributed Communication Systems

Chapter 2

Dynamic Network Topology Advances Security in Mobile Ad Hoc Networks

2.1 Introduction

Ad hoc networks eliminate the need for online infrastructure by relying on the mobile wireless nodes to collectively perform all networking functions [3] [4] [5] [28]. The characteristics of ad hoc networks make them susceptible to numerous attacks and also challenging to secure [3] [4] [5] [28]. The wireless links are inherently vulnerable and the connectivity between the nodes is sporadic due to the error-prone and shared wireless channel. Frequent connectivity problems are also caused by node mobility and node failures resulting in an unpredictable and dynamic network topology [3] [4] [16]. These inherent vulnerabilities makes it easy for attackers to compromise the networking infrastructure in the absence of robust security mechanisms [3].

In this chapter we design a robust, secure and efficient public key management scheme suitable for stationary and low to high mobility ad hoc networks. Public-key cryptography supports other important security mechanisms, such as secure routing, by providing a set of techniques and procedures for the establishment and maintenance of keying material [11] [3].

Capkun *et al* use node mobility to the advantage of security for ad hoc networks [5]. This approach

inspired us to consider the design of mechanisms that overcome and exploit the fundamental characteristics of ad hoc networks to the advantage of security[1].

By significantly extending our analysis in [16] [13] to other security mechanisms, such as, secure routing protocols [29] [30] [31] [32], we determined that the dynamic network topology and shortcomings in wireless links are at the center of the notorious challenges of securing ad hoc networks [3] [4] [5] [28]. The dynamic network topology is due to a combination of factors such as frequent route failures and node mobility. In turn, route failures may be caused by node mobility (as nodes move out of range) and link failures as a result of wireless connectivity. The wireless links are also degraded by node mobility.

We believe that robust security mechanisms for ad hoc networks are dependent on the mechanisms' ability to exploit the dynamic network topology. Node mobility (as considered in [5] [12]) becomes a secondary focus as one of the causes of dynamic network topology. From our investigations we concluded that the dynamic network topology has the most significant affect on the routing protocol's ability to effectively discover and maintain routes in response to new route requests or existing route failures. Our proposed public key management scheme exploits the dynamic network topology and its affect on the routing protocol to the advantage of security.

Problem Statement. Considering the discussion up to this point and by significantly expanding on our initial problem statement [17] and analysis in [16], we define the problem for consideration in this chapter as follows: The challenge is to design a robust, secure and efficient authority-based public key management scheme that satisfies the following properties[2]:

- **Robustness:** Adding to the requirement of robustness defined in [16], the key management scheme (including key generation, distribution, renewal and revocation) must overcome the inherent dynamic network topology of ad hoc networks, and any of the other characteristics underlying the dynamic network topology. For example, the scheme should resist the impact of route failures and node mobility.
- **Progressive robustness:** Following from the first requirement, the key management scheme should not only overcome the impact of the dynamic network topology, but exploit it to improve performance, robustness and the security of the key management scheme. We define this property as progressive robustness. However, it is important for the scheme not to become dependent on any of the underlying characteristics, such as, node mobility. Dependence

[1] We discuss the characteristics of mobile ad hoc networks in [16].

[2] We define these properties in addition to the key management scheme requirements given in [16], such as, scalability and key freshness

on mobility is a shortcoming of [5].

- **Fully distributed:** The scheme should be fully distributed and therefore equally share the responsibility of managing security associations between nodes (during the entire life cycle of the network). This is to ensure robust security services that place the same burden on the computational, memory and energy resources of all nodes [3] [4].

- **Offline authority-based:** Following from the latter requirement, a public key infrastructure (PKI) normally needs an online certificate authority (CA) to function [11] [4] [28]. In ad hoc networks the availability of an online distributed CA, as initially defined in [3], is problematic due to the dynamic network topology [28] [5] [16]. It also provides an attacker with a convenient target to compromise the key management protocol and inherently requires a dynamic trade-off between security and availability [16]. Similar to [5], the public key management scheme should completely eliminate the online CA and only rely on an offline authority.

- **Routing independency:** The key management scheme should break the routing-security interdependence cycle [33], while ensuring network scalability. Pre-distributing keying material to all the nodes, such that security associations between all nodes will be guaranteed during network formation, trivially mitigates the routing-security interdependence cycle and make the network nonscalable; the offline trusted third party needs to engage with all nodes before the network can be formed which is not practical for many applications.

- **Efficiency:** Adding to the requirement of efficiency defined in [16], the key management scheme should reduce communication and computational overhead to have negligible impact on network performance under worst case traffic and mobility scenarios, i.e. random mobility with high, but practical, node speeds and traffic loading. The scheme should also avoid inflating the routing protocol control packets in order not to waste bandwidth.

- **On demand availability:** Following from efficiency, keying material should be available in a timely fashion when required by the routing infrastructure or by any other application. Adding to the requirement of availability defined in [16], the scheme should not introduce any noticeable networking delay in the set up of the required security associations.

- **Automated:** Certificates must be distributed (on-demand) as needed by the network (routing) layer and be transparent to the network participants, that is, the scheme should require no user involvement as this introduces inherent vulnerabilities. Unnecessary user involvement makes the scheme prone to attacks that exploit human error / limitations. User involvement

also inevitably leads to a delay in the set up of security associations resulting in a window of opportunity for attackers. User interaction is more suitable for setting up security associations on the application layer [5].

- **Integration:** The key management scheme should be easy to implement and introduce minimal changes in the underlying routing protocol, that is, integrate seamlessly with existing secure routing protocols. We also require the scheme to be integrative with any generic Intrusion Detection System (IDS) that are suitable for ad hoc networks as defined in [34]. The IDS should be the authoritative source for establishing if a node is compromise and provide input into the key revocation protocol.

- **Survivability:** The offline authority essentially controls access to the network, i.e. limits use of the network to only authorized users. Examples of such networks include users that collaborate for a common purpose such as large scale emergency response operations, vehicular communications and military missions. Adding to the requirement of survivability defined in [16], we assume that the offline authority itself or its public/private key pair may be compromised during the lifetime of the network, but this should not result in the compromise of the keying material of existing networking participants.

This chapter significantly expands on our preliminary work [16] [12] [13], considering the problem statement above. In contrast to [12] [13] that address *fully self-organized* ad hoc networks we focus here on *authority-based* ad hoc networks that will support completely different applications.

We note that the given problem is particularly hard to solve for key management meeting the new requirements while breaking the *routing-security interdependence cycle* [33]. In response to this problem we design a secure, robust and efficient, authority-based public key management scheme called AuthBasedPKM. AuthBasedPKM considers mechanisms for public key generation, distribution, authentication, renewal and revocation that satisfy the requirements above.

Our contribution: To the best of our knowledge, the fact that the dynamic network topology can be exploited to aid security in ad hoc networks, without depending on mobility, has not been discussed prior to this submission.

The remainder of the chapter is structured as follows: Section 2.2 discusses related work in support of our contribution. In Section 2.3 we present an authority-based public key establishment (APKE) protocol as a building block for the proposed public key management scheme. This protocol allows a node to negotiate a blind public/private key pair with the offline CA, prior to joining the network. Section 2.4 presents our public key management scheme in more detail. AuthBased-

PKM is broken down into an off-line (initialization) phase and on-line (post-initialization) phase. Section 2.5 discusses the security and features of AuthBasedPKM and argues how it meets the given requirements. We simulate AuthBasedPKM and compare the results against out analytical analysis. In summary, Section 2.6 concludes the chapter.

2.2 Related Work

Following from our discussion in [16], authority-based schemes that use public key cryptography normally combines an off-line authority with a distributed on-line authority. The schemes presented in [3] [35] [28] are good examples of the off-line and distributed on-line authority combination. These schemes are not designed for nodes with limited computational, memory and energy resources such as sensor networks. They can therefore take advantage of the benefits of public key cryptography such as efficient key distribution [3] [28] [11].

It is widely acknowledged that maintaining an on-line distributed certificate authority (DCA) is problematic in ad hoc networks [4] [28] [16]. Based on the studies of the author [16] [19] [18], suitable threshold crypto-systems do not exists for ad hoc networks. Luo *et al* [28] get around some of the DCA's main problems by forcing DCA servers to physically go back to the off-line authority to update their shares in the DCA's shared secret[3]. In the view of the author, [28] still fails to eliminate the disadvantages of a DCA, such as, the need to dynamically adjust the trade-off between reliability and security.

Capkun *et al* [5] proposes an *authority-based* scheme where each node is preloaded by the off-line authority with a certificate. After network formation, each node becomes its own authority domain and distributes its certificate to nodes within its transmission range. The scheme in [5] thus eliminates the problems associated with the DCA by completely removing the on-line authority altogether. The authority-based solution in [5] is mainly a mobility-assisted, key distribution scheme and do not provide all the required key management mechanisms such as certificate renewal and revocation, albeit, it is designed for a different application.

The key management scheme in [32] is designed specifically for to secure the AODV routing protocol. The nodes distribute their public keys by including them in the routing control packets. A similar approach is taken in [33]. With the large number of route requests sent by on-demand routing protocols [36], inflating the control packets (specifically route request messages) consumes

[3]We describe the main problems of the schemes based on a DCA solution in [16]

valuable bandwidth and therefore may not be a feasible solution.

Lately, advances in the field have been in the area of establishing security associations for peer-to-peer communication applications [37], [38] [39] [40] [41]. These schemes allow users physically in close proximity to share keying material or to mutually authenticate. The application of these schemes are not within the direct scope of this chapter although our scheme will work for peer-to-peer applications if a common offline authority is available. How to set up such an offline authority (or "social" CA service) it a topic for future research.

To the best of our knowledge none of the existing schemes attempt to exploit the affect of the dynamic network topology associated with ad hoc networks to the advantage of security.

2.3 Proposed Offline Authority-Based Public Key Establishment Scheme

The proposed authority-based public key establishment (APKE) protocol allows a single entity, party A, to negotiate for an authority-based public/private key pair (x_A, y_A) with a trusted third party, i.e. an *off-line* certification authority, here after referred to as the (CA). The new APKE scheme serves as a cryptographic building block in the off-line initialization phase of the proposed authority-based public key management scheme, AuthBasedPKM, presented in Section 2.4.

Peterson *et al.* [42] presents a self-certified public key establishment protocol, adapted from an integration of identity-based public keys [43], the parameter hidden blind Schnorr signature scheme [44] and self-certified public keys [45]. In this chapter an APKE protocol is proposed that leverages the scheme of Peterson *et al.* The proposed APKE scheme is based on the modified ElGamal type signature variant, presented in [12]. The main motivation for proposing the new scheme is to improve on the security of [42], enhance its efficiency and to ensure compatibility with our preliminary work (subordinate public key generation and renewal procedures) presented in [12].

The following system parameters and notations are applicable:

p, q	two large primes, such that $q \mid (p-1)$.
g	generator of the cyclic subgroup of order q in $(Z)_p^*$.
$H(\cdot)$	collision free one-way hash function.
x_P	private key of party P.

y_P public key of party P, where $y_P = g^{x_P} \ mod \ p$.

To ensure the integrity of the data exchanged between the CA and party A, a secure side channel (out-of-band mechanism) is required such as an infrared interface or physical wire. The existence of such an authentic channel between party A and the CA is a reasonable assumption [28] [5].

It is assumed that the CA has a long-term public/private key pair (x_{CA}, y_{CA}), generated by choosing a random number $x_{CA} \in_R [1, q-1]$ as its private key and a corresponding public key computed as $y_{CA} = g^{x_{CA}} \ mod \ p$.

The proposed protocol is as follows:

1. The CA sends to party A the system parameters, p, q, g, $H(\cdot)$ and its public key y_{CA}.
2. Party A chooses random number $\overline{z}_A \in_R [1, q-1]$, computes $\overline{r}_A = g^{\overline{z}_A} \ mod \ p$ and transmits $\overline{r}_A$ to the CA.
3. The CA uses the modified ElGamal signature scheme presented in [12] to sign party A's certificate information KI_A. The CA chooses random number $k_{CA} \in_R [1, q-1]$ and computes $r_{CA} = g^{k_{CA}}$ and $r_A = \overline{r}_A r_{CA} \ mod \ p$. The certification information for party A is set by the CA as $KI_A = [ID_A \parallel ID_{CA} \parallel r_A \parallel r_{CA} \parallel CertNo_{CA} \parallel IssueDate_{CA} \parallel ValPeriod_{CA} \parallel AddInfo]$, where ID_{CA} is the identity of the CA, $CertNo_{CA}$ a unique sequence number, $IssueDate_{CA}$ the date of issuing the certificate, $ValPeriod_{CA}$ the validity period and $AddInfo$ some additional extension information.

   The CA must ensure that the user ID is unique. Note that the contents of KI_A can be altered based on the CA's certification policy. Inclusion of ID_A and r_A is however mandatory.

   The CA computes the signature parameter,

$$s_{CA} = x_{CA} + H(KI_A) k_{CA} \ mod \ q, \tag{2.1}$$

   and sends (s_{CA}, KI_A) to party A.

4. Party A can compute its private key as,

$$\begin{aligned} x_A &= s_{CA} + H(KI_A)\overline{k}_A = x_{CA} + H(KI_A)k_{CA} + H(KI_A)\overline{z}_A \\ &= x_{CA} + H(KI_A)[k_{CA} + \overline{z}_A] \ mod \ q \end{aligned} \tag{2.2}$$

   The triplet $(r_{CA}, \overline{r}_A, x_A)$, can be seen as the proxy signature of the CA on KI_A. Party A verifies the signature of the CA and calculates the corresponding public key using Equation

2.3:

$$y_A \equiv g^{x_A} = y_{CA} \cdot r_A^{H(KI_A)} = g^{x_{CA}} \cdot g^{H(KI_A)[k_{CA}+\overline{k}_A]} \; mod \; p \tag{2.3}$$

Note that the authority never learns the base private key of the user when using the APKE protocol. Section 2.5 address the security and performance of the proposed APKE protocol.

2.4 Proposed Authority-Based Public Key Management Scheme

The proposed authority-based public key management scheme for ad hoc networks, called Auth-BasedPKM, will be presented next. We first discuss the system and adversary model. Then we consider the *off-line* initialization phase, followed by the *on-line* post-initialization operation.

2.4.1 System and Adversary Model

Similar to [17] [5], we consider a fully distributed network of wireless nodes with generic medium access control (MAC) and routing mechanisms. Nodes can be stationary or move with low to high mobility speeds ($0m/s - 20m/s$). We assume that there are no pre-existing or online infrastructure and no form of *on-line* trusted authority, hence our scheme does not make use of a distributed certificate authority as proposed in [3] [35] [28]. The scheme requires an *off-line* trusted authority (TTP) or CA to initialize the nodes prior to joining the network. This assumption is consistent with existing literature [3] [28] [5] and as noted in [28] allows for a high-level of protection to secure high-value communication in, for example, military type applications or in any network which requires strong access control. The focus of this chapter is not on intrusion detection as defined by Zhang *et al* [34], hence we assume the existence of such a scheme which is a reasonable assumption.

We consider a straightforward, general adversary model, with the adversary model of [28] as a subset. An adversary is a malicious node that uses every means available to break the proposed key management scheme. Any *active* adversary can eavesdrop on all the communication between nodes, modify the content of messages and inject them back into the wireless channel. When a node is compromised all its public and private information is exposed to the adversary. In fact, the Dolev-Yao adversary model [46] is too restrictive, for example, it fails to capture information an adversary may gain from detailed knowledge of the protocols in use. We assume an active,

insider adversary. The adversary can therefore make use of all the basic network services, such as the routing infrastructure.

2.4.2 Offline Initialization Phase of AuthBasedPKM

Prior to joining the network, each node P_i, for $(1 \leq i \leq n)$, contacts the off-line trusted authority for the system parameters and negotiates with the offline TTP for an authority-based base public/private key pair (x_i, y_i). The proposed authority-based public key establishment (APKE) protocol used for this purpose is detailed in Section 2.3.

Each node generates a unique identifier ($Address_i$) that is bound to its base public key y_i as follows:

$$Address_i = H(y_i) \tag{2.4}$$

AuthBasedPKM requires $Address_i$ to be used as the node's network address or as a fixed part of the address. Note that this requirement places no constraints on the structure of the network addresses: the entire hash output, $Address_i$, can be used in networks with flat, static addresses or only a part of the output can be used in networks with dynamic addressing [12] [13].

After each node established a public/private key pair (x_i, y_i) via the APKE protocol, node P_i uses its base key pair to generate a subordinate public/private key pair (x'_i, y'_i) as follows [12] [13]:

1. P_i chooses a random number $k'_i \in_R [1, q-1]$ and computes $r'_i = g^{k'_i} \; mod \; p$.
2. P_i computes its new subordinate private key as:

$$x'_i = x_i + H(KI_i \parallel r'_i)k'_i \; mod \; q, \tag{2.5}$$

   where KI_i is the original keying information supplied by the off-line TTP (see Section 2.3) and r'_i is P_i's public commitment.

3. Finally P_i computes its corresponding subordinate public key as:

$$y'_i = g^{x'_i} = y_i(r'_i)^{H(KI_i \| r'_i)} \; mod \; p \tag{2.6}$$

To obtain an explicitly authentic key pair, P_i uses its newly obtained subordinate private key x'_i to sign its key information content, KI_i (concatenated with its subordinate public key y'_i and public

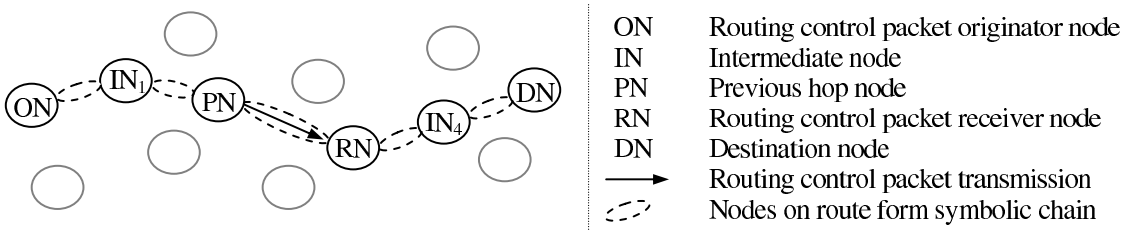

Figure 2.1: CertRelay certificate distribution main procedure

commitment β_i') via the secure ElGamal signature scheme we presented in [12] [13]. P_i's certificate can then be defined as: $Cert_i' = [KI_i \parallel y_i' \parallel CertNo_i \parallel IssueDate_i' \parallel ValPeriod_i' \parallel \alpha_i' \parallel \beta_i']$, where (α_i', β_i') is the appended signature on $(KI_i \parallel y_i' \parallel CertNo_i' \parallel IssueDate_i' \parallel ValPeriod_i' \parallel \beta_i')$.

Note that P_i's base key pair (x_i, y_i) is never used for any real communication. Rather, each P_i uses its subordinate key pair (x_i', y_i') for securing actual communication.

2.4.3 Online Post-initialization Phase of AuthBasedPKM

The post-initialization phase is defined from the start of network formation and involves mainly the following mechanisms: certificate distribution, authentication, revocation and renewal.

Each node must perform the initialization phase, as presented in Section 2.4.2, before joining the network. Only nodes with a valid subordinate certificate can join the network.

2.4.3.1 Certificate distribution

In this section we present our key distribution mechanism underlying AuthBasedPKM which plays an important part in satisfying the requirements given in Section 2.1.

AuthBasedPKM: certificate distribution.

The scheme is designed specifically to have a low implementation complexity and to allow for easy integration into most routing protocols. The proposed scheme, called Certificate Dissemination based on Message Relaying (CertRelay), is derived from the following straightforward procedure, illustrated in Figure 2.1 [17]:

When a node (RN) receives a routing control packet it checks in its certificate database if it has the certificates of the packet originator (ON) and the previous-hop node (PN) on the forward

Table 2.1: CertRelay message exchange decision table for receiver node (RN)

Case 1: ON IP address = PN IP address			
Case#	ON cert stored	PN cert stored	Actions in sequence
1a	no	no	Peer-to-Peer certificate exchange $Cert_{RN} \longrightarrow ON$;[a] $Cert_{ON} \longrightarrow RN$
1b	yes	yes	No action, process routing packet
Case 2: Originator IP address ≠ PN IP address			
Case#	ON cert stored	PN cert stored	Actions in sequence
2a	no	no	Peer-to-Peer certificate exchange $[Cert_{RN} \parallel CertQ \longrightarrow PN]$;[b] $[Cert_{PN} \parallel Cert_{ON} \longrightarrow RN]$
2b	yes	no	Peer-to-Peer certificate exchange $Cert_{RN} \longrightarrow PN$, $Cert_{PN} \longrightarrow RN$
2c	no	yes	$CertQ \longrightarrow PN$, $Cert_{ON} \longrightarrow RN$
2d	yes	yes	No action, process routing packet

[a] RN = Receiver node, ON = Originator node, $Cert_X$ = certificate of X.
[b] PN = Previous-hop node, $CertQ$ = certificate query (RN uses this message to request $Cert_{ON}$ from PN, $A \parallel B$ = concatenation of messages A and B.

route. If RN has both the certificates of ON and PN ($Cert_{ON}$ and $Cert_{PN}$), it can process the control packet as normal. If not, it requests both the certificates from PN. If RN does not have the certificate of PN it also sends its own certificate with the request to the previous-hop. Note that if RN is the first-hop on the route, then the previous-hop node and the control packet originator node will be the same entity. The routing messages thus effectively chain nodes together and allow them to relay all keying material, as required, along the virtual chains.

While reading the more detailed explanation of the proposed key relaying mechanism below, it will be useful to keep in mind an existing routing protocol. Being familiar with the operation of, for example, *endairA* [47], one of the latest *provably* secure routing protocols, will help to visualize how the proposed protocol will integrate into an existing routing protocol. We point out that any other secure routing protocol will also suffice. For example, SAODV [32] can also help to place the functionality of CertRelay into context.

The proposed key distribution scheme, CertRelay, is mainly based on the straightforward procedure introduced in Sect. 2.1. Table 2.1 explains CertRelay's core procedure in more detail from the routing control packet (RCP) receiver node's perspective (see Figure 2.1). Table 2.1 can alternatively be seen as a summary of the conditions under which the RCP receiver node (RN) will request certificates from and relay certificates to the previous-hop node (PN) in the virtual chain.

We briefly discuss Table 2.1:

- When any node in the network receives a RCP it first determines if the originator of the message (ON) has the same network address as the previous-hop node (PN) on the forward route, that is, RN has to determine if ON is the first-hop. Assume the addresses of ON and PN are equivalent as shown in Table 2.1, Case 1. RN consults its certificate repository and searches for the certificate corresponding to ON.
  - In Case 1a the search produces no result and the RN sends the ON its own certificate, $Cert_{RN}$. The ON replies with $Cert_{ON}$. After $Cert_{ON}$ is verified by RN the RCP can be processed as specified by the routing protocol.
  - If the search yields a positive result the routing message can be processed without RN requesting $Cert_{ON}$ (Case 1b).
- If the ON address and the previous-hop node (PN) address are not equal (Case 2, Table 2.1), the RN will search its certificate repository for $Cert_{ON}$ *and* $Cert_{PN}$.
  - In Case 2a the search yields a negative result. RN concatenates its own certificate $Cert_{RN}$ with a certificate query (CertQ) and relays (unicasts) the message to the previous-hop[4]. PN responds with a concatenation of its own certificate and the certificate of ON ($Cert_{PN} \parallel Cert_{ON}$). Node RN should verify both certificates before continuing to process the RCP as defined by the routing protocol.
  - If RN already has $Cert_{ON}$, but not $Cert_{PN}$, it initiates a peer-to-peer certificate exchange by sending its own certificate to PN (Case 2b). PN will respond with $Cert_{PN}$, which should be verified by RN before proceeding.
  - Case 2c is applicable if RN has $Cert_{PN}$, but not $Cert_{ON}$. This case will be the most probable since PN is within RN's local neighborhood (transmission range). RN sends PN a CertQ message. PN responds with $Cert_{ON}$. Again RN verifies $Cert_{ON}$ before processing the RCP.
  - The routing message can be processed as normal in Case 2d, since $Cert_{ON}$ and $Cert_{PN}$ are already stored in the node's certificate repository.

We have discussed how the proposed key distribution scheme can be integrated into most secure routing protocols. In summary, any routing message that is received by a node acts as a trigger for

[4]RN sends its own certificate to PN, since PN may require $Cert_{RN}$ when routing control messages are sent back via the established route. In addition, since RN and PN are neighbors they will most probably require each others certificates during future route discovery procedures. We show in Section 2.5.2.3 that the success rate of localized peer-to-peer certificates exchanges are high, thus if RN does not have $Cert_{PN}$ then PN will also not have $Cert_{RN}$ with high probability.

the node to request from the previous hop, the relaying of required keying material. The conditions that warrant the requests, and specifies the format of the requests, are defined by the rules in Table 2.1. In Section 2.5 we will analyze CertRelay in terms of efficiency and security.

2.4.3.2 Certificate authentication

Before keying material can be used to support other security mechanisms, such as message authentication, the certificates of participants should be validated.

AuthBasedPKM: certificate authentication.

Users that receive certificate $Cert_i$ have to verify the authenticity of the certificate by checking the following:

- The base public key y_i was established via the help of the off-line TTP. If Equation 2.3 holds then y_i is *implicitly* authentic.
- The subordinate public key y_i' is *implicitly* authentic if Equation 2.6 holds
- The subordinate public key y_i' and base public key y_i are explicitly authentic if appended signature (α_i', β_i') on $Cert_i$ is validated.
- In order to prevent address spoofing users should also check that $Address_i = H(y_i)$ holds.

2.4.3.3 Certificate revocation

Certificate revocation poses a challenging problem in ad hoc networks [28]. As noted in [28] most solutions, such as [4] [35], distribute a certificate revocation list (CRL) to other nodes. The authors of [28] argue that these proactive pushing mechanisms are inadequate since it constantly consumes network resources. Although we agree with [28] that (suspected) key compromises can happen more often in wireless networks than in conventional networks, we do not believe that it will increase traffic to such an extend that the pushing mechanisms of [4] [35] will *constantly* consume resources. If this is the case, i.e. keying material is frequently under suspision, then surely the security mechanisms implemented on the mobile devices themselves are inadequate.

Luo *et al* [28] takes on a "joint authority" approach with an online revocation authority to support the periodic update of certificates. As noted in [28], the renewal of certificates addresses only

(possible) compromised private keys, rather that keys *known* to be compromised. In order to revoke compromised certificates the system requires an intrusion detection mechanism, which is hard to achieve in ad hoc networks; current solutions are vulnerable to attack [28]. For this reason, [28] provides only interfaces for possible future suitable intrusion detection solutions (IDS).

Zhang *et al* [34] provide a realistic framework for intrusion detection in ad hoc networks. A suitable IDS will comprise of local and cooperative detection mechanisms [34]. It is evident that if a (local or cooperative) IDS detects the presence of a compromised node it has to notify all the nodes in the network as soon as possible of the intrusion in order to minimize the impact. An IDS will therefore most likely have to immediately broadcast the intrusion evidence to all network participants; it is widely known that broadcasting can be used to reach most of the nodes with high reliability [48].

Given the definition of key management [11] and as pointed out in [28], certificate revocation (retraction) should be independent of the IDS. The requirements for key management (with respect to key revocation) do not include how intrusions are detected and how all nodes are notified, but rather how the key management scheme prevents nodes from using a certificate belonging to a node marked as compromised by the IDS.

AuthBasedPKM: certificate revocation.

With reference to the discussion above, a localized authority domain approach to certificate revocation is very attractive and in our view the only way to 1) keep certificate revocation independent from the IDS and 2) eliminate any form of online authority. In the proposed scheme, AuthBasedPKM certificate revocation works as follows:

Each node is responsible to construct a local CRL which holds the certificates of the nodes that as been cited by the IDS to be compromised. This means that if the node received a revocation instruction from the (local or cooperative) IDS it will add the certificate of the compromised node to its CRL. Nodes will deny network services to nodes in their CRLs. Next, we discuss how AuthBasedPKM control the use of keying material as a requirement of key management [11].

The proposed certificate revocation scheme is based on CertRelay as discussed in Section 2.4.3.1. The certificate revocation scheme requires minor modifications to the logic that CertRelay uses to disseminate key material. The new decision tables, Table 2.2 and Table 2.3, based on Table 2.1, explain CertRelay with certificate revocation from the routing control packet (RCP) receiver node's perspective (see Figure 2.1). The modifications is due to the following requirements: Nodes RN, PN (and ON when PN = ON) need to check that the certificates of RN and PN is not within its CRL. If any of the certificates are revoked then the RCP is dropped and CertRelay

Table 2.2: Message exchange decision table for RN including certificate revocation: case 1

Case 1: ON IP address = PN IP address			
Case#	ON cert stored	PN cert stored	Actions in sequence
1a	no	no	RN: $Cert_{RN} \in CRL_{RN}$, DROP RCP, ABORT RN: $Cert_{RN} \notin CRL_{RN}$, $Cert_{RN} \longrightarrow ON$ ON: $Cert_{RN,ON} \in CRL_{ON}$, DROP RCP, ABORT ON: $Cert_{RN,ON} \notin CRL_{ON}$, $Cert_{ON} \longrightarrow RN$ RN: $Cert_{ON} \in CRL_{RN}$, DROP RCP, ABORT RN: $Cert_{ON} \notin CRL_{RN}$, process RCP
1b	yes	yes	$Cert_{RN,ON} \in CRL_{RN}$, DROP RCP, ABORT $Cert_{RN,ON} \notin CRL_{RN}$, process RCP

aborts. CertRelay thus effectively prevents the routing protocol from establishing a route via any honest node that detects a revoked certificate in the chain. Users that are denied networking services will have to visit the offline authority (e.g. network administrator) to investigate any compromise and to negotiate for a new base public/private key pair using the APKE protocol detailed in Section 2.3.

In summary, with the modified CertRelay we propose a solution to the certificate revocation problem. The solution is based on the notion that it is the responsibility of the intrusion detection system to 1) detect compromised nodes and 2) deliver intrusion detection evidence to all network participants. It is therefore not within the scope of the key management scheme to detect intrusion and to distributed intrusion related evidence. These assumptions are consistent with the definition of key management in [11]. The key management scheme has to action the certificate revocation instruction from the IDS, hence ensure that the compromised keying material as per the CRL is not used to further secure communications and hence to deny network access to compromised nodes.

2.4.3.4 Certificate renewal

In order to ensure the security of keying material and as required by other applications, certificates may need to be renewed.

AuthBasedPKM: certificate renewal

AuthBasedPKM makes use of a self-updating system based on the self-organized subordinate key renewal procedure as discussed in [12] [13]. As mentioned in Section 2.4.2, nodes do not use their base key pair for any real communication, but should derive a subordinate key pair (x'_i, y'_i) from

Table 2.3: Message exchange decision table for RN including certificate revocation: case 2

Case 2: Originator IP address ≠ PN IP address			
Case#	ON cert stored	PN cert stored	Actions in sequence
2a	no	no	RN: $Cert_{RN} \in CRL_{RN}$, DROP RCP, ABORT RN: $[Cert_{RN} \parallel CertQ \longrightarrow PN]$, PN: $Cert_{RN,ON,PN} \in CRL_{PN}$, DROP RCP, ABORT PN: $[Cert_{PN} \parallel Cert_{ON} \longrightarrow RN]$ RN: $Cert_{PN,ON} \in CRL_{RN}$, DROP RCP, ABORT RN: $Cert_{PN,ON} \notin CRL_{RN}$, process RCP
2b	yes	no	RN: $Cert_{RN,ON} \in CRL_{RN}$, DROP RCP, ABORT RN: $[Cert_{RN} \longrightarrow PN]$, PN: $Cert_{RN,ON,PN} \in CRL_{PN}$, DROP RCP, ABORT PN: $[Cert_{PN} \longrightarrow RN]$ RN: $Cert_{PN} \in CRL_{RN}$, DROP RCP, ABORT RN: $Cert_{PN} \notin CRL_{RN}$, process RCP
2c	no	yes	RN: $Cert_{RN,PN} \in CRL_{RN}$, DROP RCP, ABORT RN: $[Cert_{Q} \longrightarrow PN]$, PN: $Cert_{RN,ON,PN} \in CRL_{PN}$, DROP RCP, ABORT PN: $[Cert_{ON} \longrightarrow RN]$ RN: $Cert_{ON} \in CRL_{RN}$, DROP RCP, ABORT RN: $Cert_{ON} \notin CRL_{RN}$, process RCP
2d	yes	yes	$Cert_{RN,ON,PN} \in CRL_{RN}$, DROP RCP, ABORT $Cert_{RN,ON,PN} \notin CRL_{RN}$, process RCP

the base key pair. The secondary public key is then used for actual communication rather that the base public key. This significantly reduces the probability of a successful attack on a node's base key pair or the other cryptographic techniques in use [11]. The certificate renewal process is as follows:

Any user P_i can renew its subordinate key pair with a self-organized subordinate key renewal procedure: P_i simply chooses a new random number $k_i'' \in_R [1, q-1]$ and computes its renewed subordinate private key as:

$$x_A'' = x_A + H(KI_A \parallel r_i'')k_A'' \; mod \; q, \tag{2.7}$$

where $r_i'' = g^{k_i''}$. P_i compute its renewed subordinate public key as $y_i'' = g^{x_i''}$ and generate a new subordinate certificate $Cert_i'' = [KI_i \parallel y_i'' \parallel CertNo_i + 1 \parallel IssueDate_i'' \parallel ValPeriod_i'' \parallel \alpha_i'' \parallel \beta_i'']$.

Since nodes are responsible for their own keying material, they can renew their subordinate key pair as frequently as desired or as governed by the global certification policy implicitly dictated by the off-line authority. Nodes will however most likely renew their key pair in the following instances: when the user distrusts the subordinate public/private key pair or when the set certificate validity

period $ValPeriod_i'$ has expired.

Dealing with expired certificates: In the case of an expired certificate the renewed certificate $Cert_i''$ is distributed via the key distribution mechanism, CertRelay, proposed in Section 2.4.3.1. After a node has updated its expired subordinate public/private key pair and corresponding certificate it starts to use the new private key x_i'' for signing all routing control packets (for example route request messages). Any node that attempts to verify the signature will fail to validate the signature and check its certificate repository if the certificate has expired. Alternatively, a node can check if a certificate has expired before using the public key for verification purposes. If a certificate is found to be expired the node will request the renewed certificate via a $CertQ$ message from the previous node in the chain.

Dealing with user suspected certificate compromise: In case a user suspects its certificate to be compromised an instruction should be given the the IDS to act immediately to limit possible misuse of the private key; the IDS will most probably distribute a certificate revocation message to all nodes in the network via a reliable broadcast mechanism as proposed in [48]. We propose that the certificate revocation message should be flagged and CRLs should be updated with the subordinate certificate and not the base key pair to allow the user to renew the subordinate key pair. If a suspected certificate is found to be revoked then the node will request the renewed certificate via a $CertQ$ message from the previous node in the chain.

2.5 Discussion on the Security and Features of AuthBased-PKM

In this section we analyze the proposed scheme against the requirements given in Section 2.1.

- **Robustness and routing independent:** The proposed scheme overcomes the inherent dynamic network topology by making the underlying certificate distribution protocol integrated, but independent from the routing infrastructure. AuthBasedPKM does not rely on the routing scheme to provide it with routes to distribute certificates. Key material is relayed along a virtual chain using one-hop communication only. We show in Section 2.5.2.3 that the one-hop certificate exchange success rate varied between 86 % and 97 % which confirm the effectiveness of localized communication in wireless networks. If a link fails and CertRelay cannot exchange certificates between a node pair, the route would also have failed with a high probability. The routing protocol is designed to overcome this problem by design and

we exploit this inherent capability to achieve robustness for our scheme.

- **Progressive robustness:** As discussed above the key management scheme overcome the impact of the dynamic network topology by exploiting the routing protocol's inherent ability to respond to route failures. By analyzing our simulation (discussed in Section 2.5.2.3) we observed that higher node mobility and the error-prone, shared wireless channel result in more route failures as a result of various factors such as link failures, nodes moving out of range, traffic congestion, data errors etc. All of this causes route failures. The routing protocol responds by setting up a new route which triggers our certificate distribution mechanisms to set up security associations as required by the routing protocol. CertRelay therefore exploits the affect of these characteristics on the network topology to the advantage of security.
- **Fully distributed and offline authority-based:** From the description of the key management scheme in Section 2.4 it is evident that each node is its own authority domain and perform exactly the same function. The scheme is fully distributed and there is no convenient point of attack such as a distributed online CA.
- **On demand availability and automation:** The certificate distribution mechanisms is triggered by the routing protocol and the end result is that all the certificates required to secure the route is readily available. From the discussion in Section 2.4.3.1 it is clear to see that two nodes that want to securely communicate would have automatically exchanged certificates in order to secure the route. The keying material is therefore immediately available to secure any further communication and does not need any user involvement.

Next, we discuss the security/survivability of AuthBasedPKM in Section 2.5.1, followed by its performance in Section 2.5.2. The remaining requirements such as **abstraction**, **efficiency** and **key freshness** are also discussed in the following sections.

2.5.1 On the Security of AuthBasedPKM

Considering the proposed scheme's system and adversary model, given in Section 2.4.1, it becomes clear that an adversary will attack the scheme mainly in the following ways:

1. AuthBasedPKM allows authorized users to negotiate with an off-line authority for an authority-based public/private key pair using the authority-based public key establishment (APKE) protocol proposed in Section 2.3. An adversary may attempt to forge an authority-based public key, satisfying Equation 2.3, but will also have to provide a knowledge proof of the

corresponding private key. Forgery will allow an adversary to either spoof the identity of a legitimate user or join the network as a perceived "authorized" entity. See Section 2.5.1.1 for further discussion on the security of the APKE protocol.

2. From Equation 2.6, an adversary can attempt to forge a subordinate public key y_i'' that satisfies $y_i'' = g^{x_i''} = y_i(r_i')^{H(KI_i')} \ mod \ p$. We provide a strong security argument for subordinate public keys in [12]. The philosophy behind the security argument is well described by Koblitz and Menezes [49].

3. The sub-ordinate public key generation scheme presented in Section 2.4.2 only provides *implicitly* authentic public keys. Users generate subordinate certificates to bind their key pairs to their keying information KI_i as defined in Section 2.3 and to prove knowledge of the subordinate private key which result in an *explicitly* authentic public key. An adversary may attempt to forge a subordinate certificate by binding a forged subordinate public key y_i'' to the users valid keying information KI_i. Forging a subordinate certificate is however unlikely as the modified ElGamal type signature scheme used to generate a signature over the content of the certificates are formally proven secure in Random Oracle and Generic Model (ROM+GM) in [12].

4. In the post-initialization phase users exchange their certificates on a need-to-know basis using CertRelay, presented in Section 2.4.3.1. An adversary may attempt to:

   (a) Preventing users from exchanging certificates on a need-to-know basis (with the possible motivation to prohibit route discovery). We note that AuthBasedPKM will not degrade the security of the routing infrastructure, but may result in the failure of the routing mechanism if certificate exchanges initiated by the route control messages are unsuccessful. This is however not a problem since an adversary that is able to force certificate exchange failure by disrupting the channel, for example, through jamming the medium or by sinking packets by perpetrating a man-in-the-middle (MITM) attack [50] can just as well disrupt the routing protocol by blocking the routing control messages.

   (b) Trigger false or erroneous certification exchanges by sending "bogus" or unauthorized routing control messages or certificate queries. These certificate exchanges will ultimately be to the benefit of security. Bogus messages are defined as an authentic message with incorrect contents which can be sent by either a corrupt authorized user or a compromised authorized node. Unauthorized messages are normally sent by an active adversary who injects messages into the communication channel. See Section 2.5.1.3 for a discussion on the security of the certificate distribution mechanism.

2.5.1.1 On the security of the proposed authority-based public key establishment (APKE) scheme

We refer to the combined security model, the Random Oracle and Generic Model (ROM+GM), proposed by Schnorr *et al.* [51] [52]. The security of the proposed APKE scheme, proposed in Section 2.3, can be formally proven against one-more forgery in ROM+GM. As described by Schnorr *et al.* we also assume an interactive, generic adversary $\mathcal{A}$.

Given the system parameter setup in Section 2.3, resilience against the one-more forgery attack prevents $\mathcal{A}$ from performing t generic steps, which includes l sequential interactions with an off-line (certificate) authority, to produce $l+1$ valid authority-based public/private key pairs with probability better than $\frac{\binom{t}{2}}{q}$.

Any adversary can produce an authority-based public key y'_A that satisfies Equation 2.3 by choosing a random number $k'_A \in [1, q]$, computing $r'_A = g^{k'_A}$ and generating $y'_A = y_{CA} \cdot (r'_A)^{H(KI'_A)} \ mod\ p$. This is however not a security vulnerability since the adversary does not know x'_A where $y'_A = g^{x'_A}$. As also pointed out by Petersen *et al* [42], public keys verified by any outsider V via Equation 2.3 is only accepted as *implicitly* authentic. This implies the adversary $\mathcal{A}$ will have to provide a knowledge proof of private key x'_A to V before y'_A is taken as *explicitly* authentic. The aim of the security proof is thus to show that an adversary $\mathcal{A}$ can obtain private key x'_A that satisfies $y'_A \equiv g^{x'_A} = y_{CA} \cdot (r'_A)^{H(KI'_A)} \ mod\ p$ with only a negligible probability.

Theorem 1. *Let a generic adversary $\mathcal{A}$ interact with a off-line (certificate) authority (CA) and be given g, the public key y_{CA} and an oracle for H. $\mathcal{A}$ performs t generic steps which include l sequential interactions with the* CA. *With a probability space consisting of y_{CA}, H and the coin flips of the CA, it is not possible for $\mathcal{A}$ to produce $l+1$ authority-based private keys with a probability better than $\frac{\binom{t}{2}}{q}$.*

In the following proof *Lemma 1* and *Lemma 2* are those defined and proved in [51].

Proof. [following Schnorr *et al.* [51]]

As given by *Lemma A* defined below, the group element $f_{i'} = g^{\frac{s'_i}{c'_i}} g^{-\frac{x}{c'_i}} = g^{\langle \alpha_{\mathbf{i}'}, (1,x,\mathbf{k}) \rangle}$ for an arbitrary $i \leq t'$. $\mathcal{A}$ receives hash query $c'_i = H(m \parallel g^{\frac{s'_i}{c'_i}} g^{-\frac{x}{c'_i}})$ and needs to find s'_i which satisfies Equation 2.11. The adversary $\mathcal{A}$ is thus required to solve a linear polynomial $s'_i = x_{CA} + c'_i \langle \alpha_{\mathbf{i}'}, (1, x, \mathbf{k}) \rangle$ at $(x_{CA}, \mathbf{k})$. By *Lemma 2* presented by [51], x_{CA} is statistically independent from $(\alpha_{\mathbf{i}'}, (1, x, \mathbf{k}))$, excluding prior collisions $f_j = f_k$. By *Lemma 1* presented in [51], it is known that such collisions

will only occur with an upper bound probability of $\frac{\binom{t'}{2}}{q}$. On the other hand, by *Lemma A*, adversary $\mathcal{A}$ must choose $c_1, \ldots, c_l$ for each signature (m'_i, c'_i, s'_i) that satisfies Equation 2.8 such that x_{CA} cancels out. In the case of a sequential attack, without any collisions among the computed group elements $f_1, \ldots, f_{t'}$, the system of $l+1$ equations for $c_1, \ldots, c_l$ is solvable with an upper bound probability of $\frac{\binom{t''}{2}}{q}$, where t'' denotes the number of queries to H [51]. Equation 2.12 shows that adding $c'_i z'_i$ to s'_i has no effect on the solvability of Equation 2.8 and thus give $\mathcal{A}$ no addition advantage. It follows from $\frac{\binom{t'}{2}}{q} + \frac{\binom{t''}{2}}{q} \leq \frac{\binom{t}{2}}{q}$, that $\frac{\binom{t}{2}}{q}$ is the highest probability for $\mathcal{A}$ to succeed in a sequential, *one-more forgery attack* on the authority-based public key establishment scheme presented in Section 2.3. □

Lemma A. *Let the triplet (KI'_i, c'_i, s'_i) be an signature generated by the CA with a probability better than $\frac{1}{q}$, where $KI \in M$ in ROM. The triplet serves as intermediate keying material for the authority-based private key triplet (KI'_i, c'_i, x'_i). The c'_i-coordinate then coincides with the value $H(m \parallel f)$ corresponding to the hash query $(m \parallel f)$. From Equation 2.1, $g^k = g^{\frac{s}{c}} g^{-\frac{x_{CA}}{c}}$. The hash query $(m \parallel f) \in G \times M$, satisfies $c'_i = H(m \parallel f) = H(m \parallel g^{\frac{s'_i}{c'_i}} g^{-\frac{x_{CA}}{c'_i}})$, where the group element $f = f'_i$ for some arbitrary $1 \leq i' \leq t'$. As shown in Equation 2.2 the adversary $\mathcal{A}$ adds the product of a random number (z'_i) and $H(m \parallel f)$ to produce the final authority-based private key triplet (KI'_i, c'_i, x'_i).*

The parameters (KI'_i, c'_i, s'_i) also satisfy:

$$c'_i = \frac{1}{-\alpha_{i',1} + \sum_{k=1}^{l} \left[\alpha_{i',k}\, c_k^{-1}\right]} \tag{2.8}$$

$$s'_i = c'_i \left[\alpha_{i',0} + \sum_{k=1}^{l} \alpha_{i',k} \frac{s_k}{c_k}\right], \tag{2.9}$$

while (KI'_i, c'_i, x'_i) satisfies Equation 2.8 and the following:

$$x'_i = c'_i \left[\alpha_{i',0} + \sum_{k=1}^{l} \alpha_{i',k} \frac{s_k}{c_k} + z'_i\right] \tag{2.10}$$

In the following proof, *Lemma 2* is as defined and proved in [51].

Proof. [following Schnorr *et al.* [51]]

The triplet (KI'_i, c'_i, s'_i) define a signature based on equation $c'_i = H(m \parallel g^{\frac{s'_i}{c'_i}} g^{-\frac{x_{CA}}{c'_i}})$. Adversary $\mathcal{A}$ as no choice but to query hash oracle H otherwise the equality will only hold with a probability of

$\frac{1}{q}$ in ROM. Since $1 \leq i' \leq t'$ denotes the index of f among the computed group elements $f_1, \ldots, f'_t$, the group element can be written as $f_{i'} = g^{\frac{s'_i}{c'_i}} g^{-\frac{x_{CA}}{c'_i}} = g^{\langle \alpha_{i'}, (1, x_{CA}, \mathbf{k}) \rangle}$. This follows from the fact that $\mathcal{A}$ computes group elements $f'_i = g^{a_{i,0}} y_{CA}^{a_{i,1}} g^{k_1 a_{i,2} + \cdots + k_l a_{i,l+1}} = g^{\langle \alpha_{i'}, (1, x_{CA}, \mathbf{k}) \rangle}$ in the *generic* model. It follows from the previous equations and $k_k = \frac{s_k}{c_k} - \frac{x_{CA}}{c_k}$, that:

$$s'_i = x_{CA} + c'_i \log_g \left[g^{\frac{s'_i}{c'_i}} g^{-\frac{x_{CA}}{c'_i}} \right] = x_{CA} + c'_i \langle \alpha_{i'}, (1, x_{CA}, \mathbf{k}) \rangle \tag{2.11}$$

$$\begin{aligned} s'_i &= c'_i \left[\alpha_{i',0} + \sum_{k=1}^{l} \alpha_{i',k} \frac{s_k}{c_k} \right] + \\ & \quad x_{CA} \left[1 + c'_i \left[\alpha_{i',1} - \sum_{k=1}^{l} \alpha_{i',k} \frac{1}{c_k} \right] \right] \end{aligned} \tag{2.12}$$

In order for the generic adversary $\mathcal{A}$ to calculate the correct s'_i, $\mathcal{A}$ must find c'_i such that x_{CA} cancels out. $\mathcal{A}$ must therefore select $c_1, \ldots, c_l$ that satisfies Equation 2.8.

If x_{CA} cancels out, x'_i can be computed by $\mathcal{A}$ as specified by Equation 2.10.

In the case that x_{CA} does not cancel out in the equality given by Equation 2.12, the equality will only hold with probability $\frac{1}{q}$ since x_{CA} is statistically independent from non-group data by *Lemma 2* presented in [51].

Adversary $\mathcal{A}$ adds $c'_i z'_i$ to s'_i which yields:

$$x'_i = x_{CA} + c'_i \log_g \left[g^{\frac{s'_i}{c'_i}} g^{-\frac{x_{CA}}{c'_i}} \right] + c'_i k'_i = x_{CA} + c'_i \langle \alpha_{i'}, (1, x_{CA}, \mathbf{k}) \rangle + c'_i z'_i \tag{2.13}$$

$$\begin{aligned} x'_i &= c'_i \left[\alpha_{i',0} + \sum_{k=1}^{l} \alpha_{i',k} \frac{s_k}{c_k} + z'_i \right] + \\ & \quad x_{CA} \left[1 + c'_i \left[\alpha_{i',1} - \sum_{k=1}^{l} \alpha_{i',k} \frac{1}{c_k} \right] \right] \end{aligned} \tag{2.14}$$

□

2.5.1.2 On the security of subordinate public keys and hash based identifiers

The security analysis of the subordinate public key generation scheme and hash based identifiers (Section 2.4.2), as discussed in [12] [13], is repeated here for the sake completeness:

Subordinate public keys

From any entity's perspective, Equation 2.6 can only provide *implicit* authentication for subordinate public key y'_i, i.e. the verification procedure gives no assurance that P_i knows the corresponding private key x'_i. The authenticity of the subordinate public key only becomes explicit when P_i uses it for a cryptographic procedure which inherently provides a proof of knowledge of x'_i.

An adversary $\mathcal{A}$ that wants to produce a forged subordinate public key must compute a public key $y'_{\mathcal{A}}$ that satisfies:

$$y'_{\mathcal{A}} = y_i \cdot (r_{\mathcal{A}})^{H(KI_{\mathcal{A}})} \; mod \; p \tag{2.15}$$

$\mathcal{A}$ does not know $\log_g y'_{\mathcal{A}}$ and will consequently fail to produce a valid signature that satisfies Equation 2.6. This serves as motivation for introducing subordinate certificate generation in Section 2.4.2, which allows the subordinate public keys to be explicitly authenticated. It will thus be appropriate to assess the security of the proposed subordinate public key generation protocol in conjunction with the signature (α'_i, β'_i) on $m_i = [KI_i \parallel y'_i]$ as described in Section 2.4.2. It is noted that (α'_i, β'_i) is produced via the secure signature scheme presented in [12] [13]. The verification equation on (α'_i, β'_i) is given as:

$$g^{\alpha'_i} = y'_i \cdot (\beta'_i)^{H(m_i \| \beta'_i)} \; mod \; p \tag{2.16}$$

Substituting Equation 2.6 into Equation 2.16 yields:

$$g^{\alpha'_i} = y_i \cdot (r_i)^{H(KI_i)} \cdot (\beta'_i)^{H(m_i \| \beta'_i)} \; mod \; p, \tag{2.17}$$

which has the following signature equation:

$$\begin{aligned} \alpha'_i \;\; &= \;\; x_i + H(KI_i)(k_i) \\ &\quad + H(m_i \parallel \beta'_i)(\log_g \beta'_i) \; mod \; q \end{aligned} \tag{2.18}$$

An entity which explicitly authenticates y'_A via Equation 2.6 and Equation 2.16, indirectly verifies

Equation 2.18 in two steps. From Equation 2.17 and Equation 2.18 it is concluded that an adversary $\mathcal{A}$ can only generate a forged subordinate public key with an upper bound probability of $\frac{\binom{t}{2}}{q}$ in the ROM+GM security model (as per *Theorem 1* [12] [13]). Furthermore it shows that the verifier of Equation 2.6 and Equation 2.16 can be assured that the party with ID_i, generated via Equation 2.4, knows the base private key x_i corresponding to y_i.

Another point of concern is that a compromise of the subordinate private keys may reveal information about the base or primary private key. From Equation 2.5 it can be seen that the base private key x_i' is blinded from the subordinate private key x_i' by the addition of a random number k_i'. This is the same mechanism that is used by all ElGamal type signatures to protect private keys from being derived from valid signatures [53]. An adversary $\mathcal{A}$ that compromises a subordinate key pair (x_i', y_i') therefore has the same probability of gaining knowledge of the base private key x_i as someone with a valid signature ElGamal signature, generated via the signature scheme presented in [12] [13].

Hash based identifiers

The security of crypto-based identifiers has been extensively reviewed in [54] [55] [56] [33]. As noted from the specification of the proposed key management scheme, AuthBasedPKM is not bound to any specific hash function. Such a secure hash function can be carefully chosen on deployment of the protocol. For example, currently SHA-1 [57] with a 160-bit output will provide adequate security since 2^{80} hash operations are needed to find a collision on SHA-1 using a brute-force attack. The most recent known attack on SHA-1, presented in [58], shows that a collision on SHA-1 can be found with a complexity of less than 2^{69} hash operations. Note however, that an attacker also has the additional computational overhead of one full exponentiation in order to find a valid public key before computing the hash function. Clearly the additional time complexity of the exponentiation makes the spoofing of network addresses impractical [12] [13].

2.5.1.3 On the security of CertRelay

The security of the key distribution scheme presented in Section 2.4.3.1 will be discussion next:

To ensure the integrity of all messages sent by CertRelay we require that messages are signed using a secure digital signature scheme (for example RSA). Ideally CertRelay should use the same signature scheme as deployed by the underlying routing protocol. A unique sequence number or random number (to guarantee the uniqueness of each message) must also be included in the

messages to avoid replay attacks.

In the remainder of the section we will analyze the security of CertRelay in the authenticated-links adversarial model (AM) of Bellare, Canetti, and Krawczyk [59]. Cagalj, Capkun and Hubaux [50] also uses AM to prove the security of their scheme, which supports our use of AM. As formally proven in [59] and further explained in [50], a strong security argument in the AM model (or ideal world model) will also apply in the unauthenticated links model (UM) by correctly applying a signature-based *message transmission* (MT)-authenticator to each message sent. The security of the protocol, if provably secure in an *authenticated* network, can then be conveniently reduced to the security of the digital signature scheme in an *unauthenticated* network [59]. The goal is thus to show that CertRelay is secure in AM, which will imply equivalence in UM. Without losing credence in the security argument we will keep our treatment informal, but firmly rooted in the formal foundations of the AM adversarial model defined by [59].

Consider Case 1a in Table 2.1, which portrays a generic communication scenario in CertRelay. The discussion also applies with minor modifications to any of the other cases (Case 1b to 2d). Let ON be party A and RN party B [5]. Note that in Case 1a the originator node is the same entity as the previous-hop node (ON = PN). An AM adversary ($\mathcal{M}$) models the authentication protocol executed by party A and party B (from A's perspective) as an oracle $\prod_{A,B}^{s}$ with session ID $s \in \mathbb{N}$ [60]. In the same way, queries sent to B from $\mathcal{M}$ and the corresponding responses are modelled by oracle $\prod_{B,A}^{t}$, where session ID $t \in \mathbb{N}$. Using the notation of [50], the timely messages sent to and received from $\prod_{A,B}^{s}$ are denoted by conversation $conv_A$ and $conv_B$ for $\prod_{B,A}^{t}$. Oracles $\prod_{A,B}^{s}$ and $\prod_{B,A}^{t}$ have *matching conversations* (as defined in [60] and further explained in [50]) if message m sent out by $\prod_{A,B}^{s}$ at time τ_i is received by $\prod_{B,A}^{t}$ at time τ_{i+1}.

In the AM model the adversary $\mathcal{M}$ has full control, that is, $\mathcal{M}$ can activate or corrupt parties at random, but cannot forge or replay messages to impersonate uncorrupted parties and is also bound to deliver sent messages faithfully [59]. The CertRelay CertRelay protocol commences by $\mathcal{M}$ activating $\prod_{A,B}^{s}$ at time τ_0. The outgoing routing control message R_{msg} of contains the identity (or network address) of A [6]. The AM adversary cannot modify the network address (identity) in the AM model by definition (see [59]) and has to deliver the message to $\prod_{B,A}^{t}$, modelling an arbitrary party B of $\mathcal{M}$'s choice[7]. Incoming message R_{msg} activates $\prod_{B,A}^{t}$ to respond with B's certificate $Cert_B$ at time τ_1 (any other activation will not comply with CertRelay). $Cert_B$ (containing the

[5] We assume that both A and B can be trusted to behave as specified by CertRelay, otherwise there is not much to discuss.

[6] Once $\mathcal{M}$ has activated $\prod_{A,B}^{s}$ for an arbitrary party A, $\mathcal{M}$ cannot alter the identity of A anymore without violating CertRelay or the rules of AM.

[7] Party B does not necessarily know the identity of A *a priori* and does not need to until B receives R_{msg} from A. If B receives the R_{msg}, $\mathcal{M}$ cannot alter the identity of B anymore without violating CertRelay or AM.

identity of B) is appended to A's identity and delivered to $\prod^{s}_{A,B}$ as required of $\mathcal{M}$. Up to this point there is not much the adversary can do to attack the protocol; according to the definition of AM, $\mathcal{M}$ can activate any of the oracles (in an appropriate manner in compliance with the CertRelay protocol), but cannot forge messages coming from the oracles that simulate uncorrupted parties (A and B) and has to deliver the outgoing messages after activation to the oracles. In the next round the AM adversary has no option but to activate $\prod^{s}_{A,B}$ which will respond to $\prod^{t}_{B,A}$ with $Cert_A$ (containing the identity of A, appended with the identity of B) at time τ_2. Since $\tau_0 < \tau_1 < \tau_2 < \tau_3$ and $conv_A$ and $conv_B$ are matching conversations, as illustrated below, both oracles will output "Accept" [8].

$$\begin{aligned} conv_A &= (\tau_0, \perp, R_{msg}), (\tau_2, CertB, CertA); \\ conv_B &= (\tau_1, R_{msg}, Cert_B), (\tau_3, CertA, \perp); \end{aligned}$$

As described above the AM adversary cannot attack CertRelay in the AM model without breaking the rules of AM or modifying the oracles not to comply with CertRelay. Considering the communication model of [59] and the security argument above, it is clear that CertRelay is a *message driven protocol* (as defined in [59]) that forces *matching conversations* between parties that engage via CertRelay. CertRelay is therefore a secure mutual authentication protocol (with authenticated data as described by [60]) in the AM model:

As mentioned above CertRelay can be transformed from a secure AM protocol to a secure UM protocol using a signature-based MT-authenticator [59]; each unique message m (containing the identity of the sender) is signed with the private key of the sender. The signatures are verified with the sender's corresponding public key. Each public key is bound to the identity of the corresponding private key holder by an offline authority to form a certificate. As assumed in the system model (see Section 2.4.1) each network participant has the authentic public key of the offline authority readily available to verify the authenticity of the received certificates. Successful verification convinces the receiver of the binding between the public key and the user's identity (network address). Since the certificates are included in the exchanged messages, it is therefore clear that CertRelay is a mutual authentication protocol in the UM model with an exchange of *implicitly* authenticated data. As a final observation we note that the probability of *No-Matching*, as defined in [60], between $conv_A$ and $conv_B$ (in the UM model) is given by the probability that the adversary can break the underlying signature scheme, which should be negligible if the signature scheme is carefully chosen and securely implemented.

[8] To remain compatible with [50] we also use $\perp$ to denote that a party receives/sends no message in the corresponding time τ_i.

2.5.2 On the Performance of AuthBasedPKM

First we analyze the efficiency of AuthBasedPKM for the initialization (online) and post-initialization (offline) phases. In Section 2.5.2.3, we evaluate the performance of AuthBasedPKM in a simulation study.

2.5.2.1 Efficiency of AuthBasedPKM initialization phase

The initialization phase is performed by each node before joining the network and therefore has no impact on network performance. This process should however still be as efficient as possible. Each node P_i performs 4 exponentiations (exp), 3 random number generations (R_{gen}) and 3 hash computations ($H(\cdot)$) (The 2 multiplications and 2 summations have insignificant impact on the time complexity in comparison with the exponentiations). The initialization phase has no communication cost. The user only needs to visit the offline authority once before joining the network and will not experience any noticeable time delay due to the cryptographic initialization operations.

2.5.2.2 Efficiency of AuthBasedPKM post-initialization phase

The post-initialization phase of AuthBasedPKM is defined by the certificate distribution scheme, called CertRelay (see Section 2.4.3.1). The efficiency analysis of CertRelay in an ideal setting, i.e. assuming guaranteed connectivity, is rather easy. Certificate exchanges all take place on a peer-to-peer basis. From Table 2.1 it can be seen that all the exchanges take at most two asynchronous rounds with one unicast message from each node. Each node pair only exchanges their certificates once on a need-to-know basis. Although it is possible to derive an analytical model to evaluate the performance, such a model will be of limited use as it will be dependent on many assumptions related to the wireless channel, MAC protocol, routing protocol, mobility model etc. In the following section we evaluate CertRelay in a more realistic setting using simulations as commonly done in the evaluation of ad hoc network protocols.

2.5.2.3 Simulations

The performance of AuthBasedPKM was evaluated in a simulation study where the dynamic network topology caused by factors such as poor connectivity and route failures (due to the error-prone

wireless channel, node mobility, traffic congestion etc.) have an impact on the network operation. The ease of coding AuthBasedPKM in the ns-2 simulator [61] confirmed the low implementation complexity of the proposed key management scheme.

Simulation model: In the simulation of AuthBasedPKM we used the IEEE 802.11b physical layer and medium access control (MAC) protocols included in the ns-2 simulator. The radio-model was set to a nominal bit-rate of 11Mb/s and a transmission range of 250m. The network area for all simulations was set to 1000m x 1000m. The ns-2 constant bit-rate (CBR) traffic generator was used to set up the connection patterns. For all simulations a 512byte CBR packet size was used and the traffic loading was varied between 1 CBR packet/sec and 7 CBR packets/sec. The size of certificates was also set to 512 bytes. The total of 50 nodes, each had one CBR traffic connection with a single unique destination. The traffic sources were started within the first 60sec of each 1000sec simulation. We note that this is unlikely to occur in practice, but it is an effective strategy to force as much certificate distribution activity as possible from the start of network formation.

The choice of an appropriate mobility model is a problem and it is unlikely that everybody will agree with any specific choice. Although mobility models for ad hoc networks have received much attention lately [62], a widely used, "realistic" mobility model is not available and it is unlikely to appear due to the application specific nature of mobility patterns. To be consistent with most literature the random waypoint model was chosen to simulate node mobility. It can be argued that such a model does not always reflect reality. For example, [5] presents a mobility model which more "accurately" reflects how nodes would move in reality. This model is referred to as the *Restricted Random Waypoint* mobility model [5]. Rather than choosing its destination as a random point on the plane, a node chooses a destination point from a finite set with probability ρ and a random point with probability $1-\rho$. This model clearly fuels the rate that nodes come within "close" range since they move towards common meeting points with a higher probability. Nevertheless, to be consistent with most literature, *mobgen-ss* [63] was chosen as a mobility scenario generator based on the random waypoint model. It is pointed out that the *setdest* mobility generator included in the ns-2 distribution is flawed [63]. The initial probability distribution of *setdest* differs at a later point in time as it converges to a "steady-state distribution" [63].

We wanted to observe the effectiveness of CertRelay at very low (almost stationary), moderate and high node mobility. In the simulations the mean speed was set to 0.1m/sec, 5m/sec and 20m/sec for each traffic scenario. These mobility speeds are widely used in ad hoc network simulations based on the random waypoint model. Since a pause time greater than zero reduces the relative node speed, the pause time was set to zero.

The Ad Hoc On-demand Distance Vector (AODV) routing protocol [64] was chosen for the simulations. The implementation of CertRelay in CertRelay in ns-2 closely followed the discussions in Section 2.4.3.1 and will not be explained here in repetition.

We limited the scope of our analysis to the routing and upper layers; we do not consider the message overhead occurred on the lower layers as the effect of this will manifest in the upper layers.

Simulation results: Next, we present the simulation results of AuthBasedPKM. The aim is to make an assessment of AuthBasedPKM's impact on network performance. The following two metrics are observed: 1) Constant bit-rate (CBR) packet delivery ratio (PDR) as a function of mobility and load. 2) CBR packet end-to-end delay as a function of mobility and load.

In Figure 2.2, it can be seen that the PDR of the "CBR reference" simulation corresponds closely with that of the "CBR with CertRelay" simulation. In fact, we can claim that the impact on network performance is negligible for 0.1m/sec, 5m/sec and 20m/sec mobility. As per design specification, AuthBasedPKM exploits the dynamic network topology; as the mobility increases and the topology changes more rapidly the CBR and "CBR with CertRelay" simulations become even more correlated (see Figure 2.2). The mobility characteristic of ad hoc networks is widely regarded as a limiting factor, as it is a major contribution to route failures and hence changes in network topology. The close relation between the CBR and "CBR with CertRelay" at 0.1m/sec indicates that AuthBasedPKM in contrast to previous efforts [5] does not rely on mobility. We believe that [5] mainly indicates that mobility can aid security on the application layer. We thus make a novel contribution and show that the dynamic network topology (caused by mobility among other factors) can aid security on the routing layer *without* forcing security to depend on mobility.

To place the PDR vs. load results into context, the average CBR packet end-to-end delay is shown in Figure 2.3. The figure confirms that CertRelay does not add any significant delay to the delivery of CBR packets for 0.1m/sec, 5m/sec and 20m/sec mobility.

We further investigate AuthBasedPKM's negligible impact on network performance. Figure 2.4 shows the cumulative number of certificates sent by AuthBasedPKM vs. simulation time. As per the simulation model all CBR flows were started within the first 60sec of the 1000sec simulation. The objective was to stimulate as many certificate exchanges as possible in order to establish the worst case impact of the proposed certificate distribution on network performance. Figure 2.4 indicates that there is a flurry of certificate distribution in the first 100sec, after which the graphs level off. This means that our scheme can set up almost 100 % of the security associations in under 100 sec. Closer inspection reveals that the rate of certificate distribution increases with

mobility. We can also see that the 20m/sec plot remains almost completely level after 100sec while there is still some certificate exchange activity after 100sec in the 0.1m/sec and 5m/sec mobility scenarios. Our investigations into the ns-2 trace files have shown that with 20m/sec mobility most certificates are exchanged on a peer-to-peer basis as the nodes roam the network. As the mobility decreases the availability of the routing control packet originator node's certificate decreases and more certificates are being relayed along the chains during the remaining duration of the simulation. Figure 2.4 indicates that an increase in mobility fuels the rate at which security associations are established, and also decreases the total number of certificates distributed during the simulated timeframe. From the simulation results it is also clear that higher mobility causes more route failures, which results in more route request messages transversing the network. As more route requests are being broadcast, more localized certificate exchanges are initiated which increases the rate at which security associations are being established.

AuthBasedPKM avoids dependence on mobility by using only localized (one-hop) communication. Certificates needed from nodes not within the transmission range are relayed along the chain formed by intermediate nodes. The effectiveness of this mechanism relies on the node's channel access success rate, which is MAC protocol specific. Figure 2.5 shows that this form of communication with the IEEE 802.11b MAC protocol is very effective; between 86% and 95% of the certificates sent between nodes on a one-hop basis are delivered. As the load increases one would expect a significant decrease in the certificate delivery ratio. What we can see from Figure 2.5 is that the certificate delivery ratio does decrease with an increase in mobility but does not deteriorate significantly as the load increases. Considering Figure 2.4 and comparing Figure 2.3 with 2.5 show that the performance of AuthBasedPKM improves relatively to the CBR packet delivery ratio as the network topology becomes more dynamic. This confirms that the dynamic network topology of ad hoc networks can be used to advance security. This also substantiates our claim that the proposed certificate scheme is in fact not "dependent" on the routing infrastructure's performance.

Considering the above discussions we can conclude that the message relaying mechanism of the proposed scheme is a practical way of distributing keying material in ad hoc networks with low to high mobility.

In Section 2.1 we argue that including public keys within the header of the routing control packets, as used in [32] [33], is not an efficient solution. In Figure 2.6 and 2.7 we show simulation results that support our notion. We simulated AODV with public keys of size 160 bit (20 byte), 256 bit (32 byte) and 512 bit (64 byte), included in the routing control packet headers. These public key sizes correspond to using elliptic curve cryptography with a moderate to high security level [65].

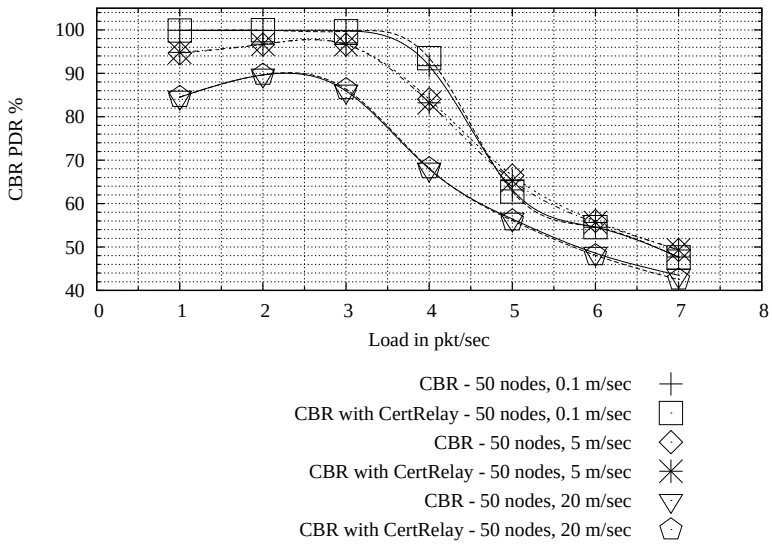

Figure 2.2: CertRelay's CBR packet delivery ratio % vs. load in pkt/sec for 0.1m/sec, 5m/sec and 20m/sec mobility

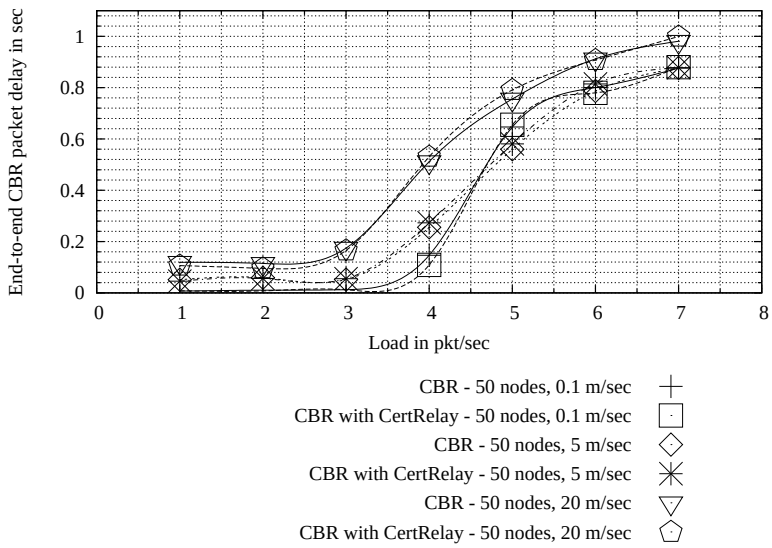

Figure 2.3: CertRelay's CBR packet end-to-end delay vs. load in pkt/sec for 0.1m/sec, 5m/sec and 20m/sec mobility

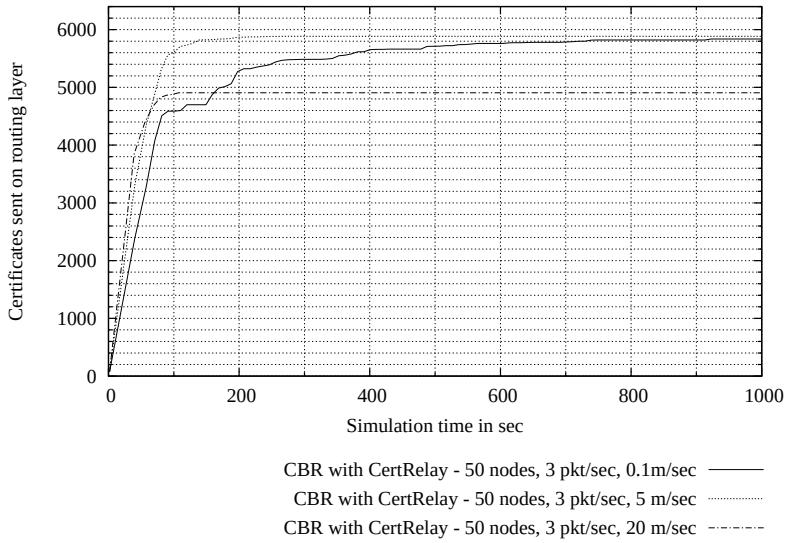

Figure 2.4: Certificates sent by CertRelay vs. time in pkt/sec for 0.1m/sec, 5m/sec and 20m/sec mobility

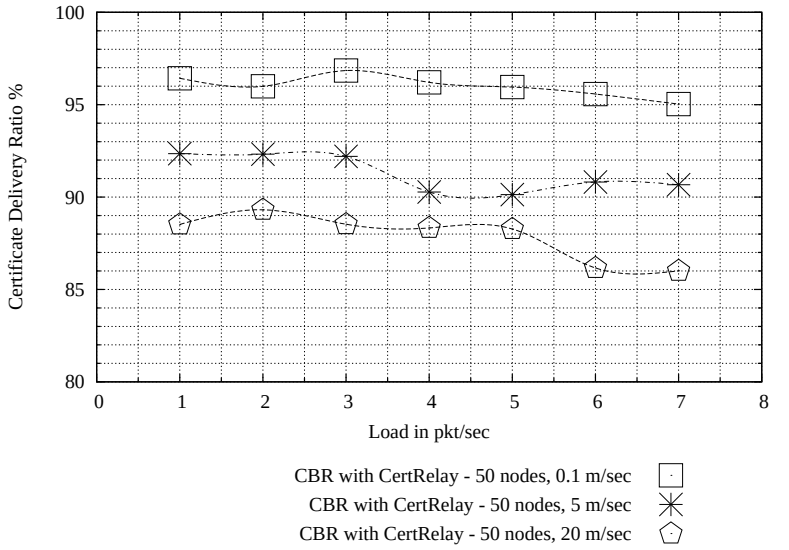

Figure 2.5: CertRelay's certificate delivery ratio % vs. load in pkt/sec for 0.1m/sec, 5m/sec and 20m/sec mobility

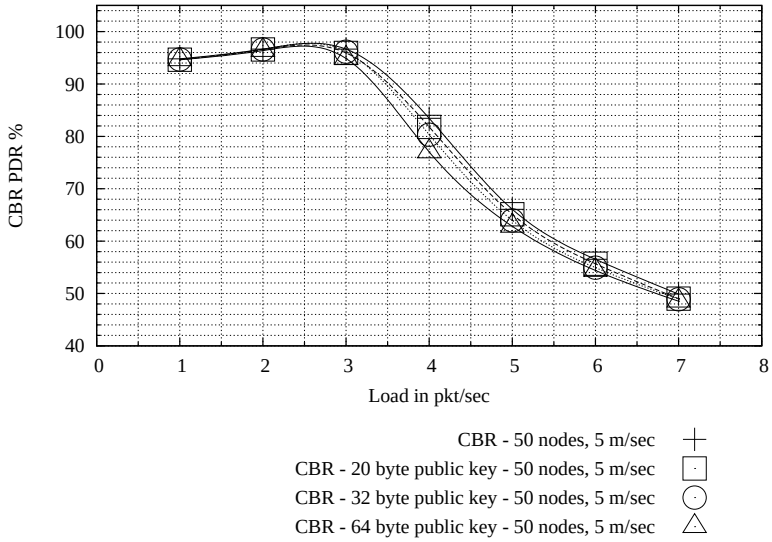

Figure 2.6: CBR packet delivery ratio % (with and without public keys in AODV headers) vs. load in pkt/sec for 5m/sec mobility

The PDR figures show that the added overhead results in approximately 6 - 8 % decrease in performance at moderate to high mobility (3/4 pkt/sec, 512 bit key). Including a more realistic average certificate of size 0.5 to 1 kilobyte (1048576 bit) in the AODV header completely degrades the network performance in the given simulation model. We deliberately kept the additions to the AODV header small to show the result of adding keying material to the RCP headers.

Considering the impact on performance in terms of CBR packet delivery ratio only gives half the picture. Figure 2.8 and 2.9 show the end-to-end delay which confirms that inflating the routing control packets comes at the price of significantly degrading the network performance. As can be seen from the plots, the end-to-end delay increases up to 0.24 sec at low or high mobility (4 - 5 pkt/sec, 512 bit / 64 byte key). We conclude that in general, as expected, the situation worsens with an increase in node mobility, but what may be surprising is the performance sensitivity due to marginal increases in the header size (for example 0 vs 20 byte included). This is explained by the high volume of routing control packets. With inclusion of an average sized, 0.5 to 1 kilobyte certificate in the AODV headers, the delay becomes impractical.

2.6 Conclusion

This chapter identifies a new problem within the area of key management and shows that a dynamic network topology can advance security for ad hoc networks. In response to the problem

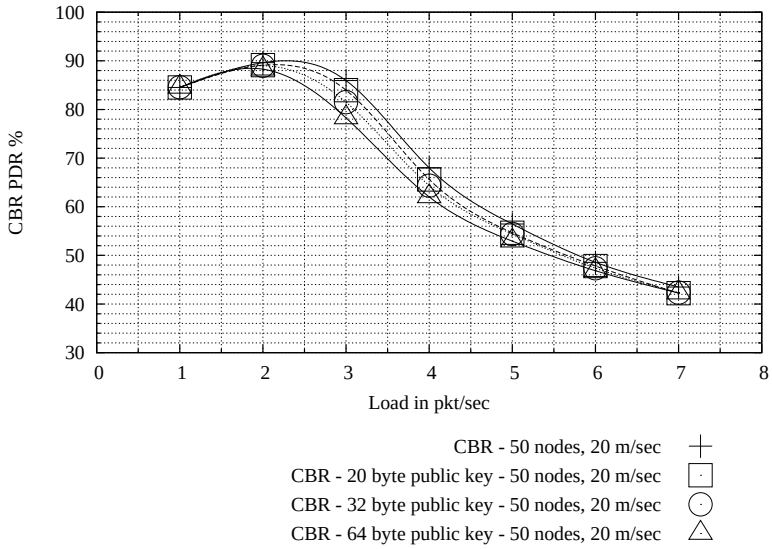

Figure 2.7: CBR packet delivery ratio % (with and without public keys in AODV headers) vs. load in pkt/sec for 20m/sec mobility

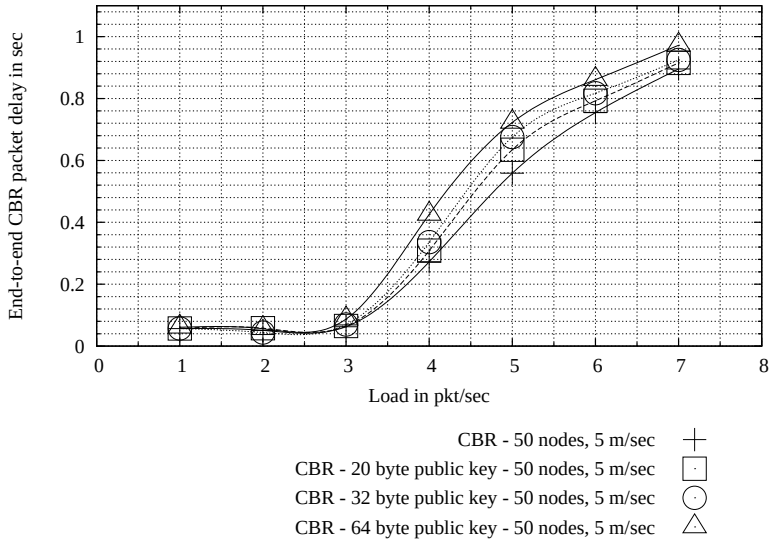

Figure 2.8: CBR packet end-to-end delay (with and without public keys in AODV headers) vs. load in pkt/sec for 5m/sec mobility

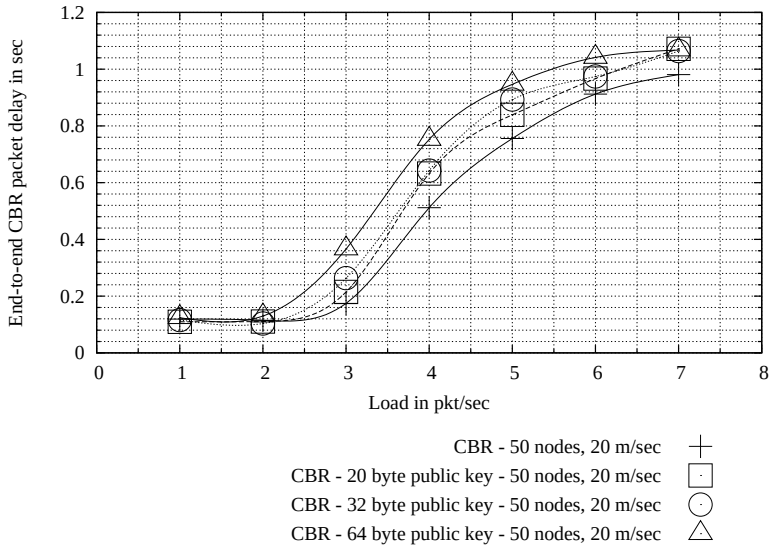

Figure 2.9: CBR packet end-to-end delay (with and without public keys in AODV headers) vs. load in pkt/sec for 20m/sec mobility

we contribute a novel authority-based public key management scheme, called AuthBasedPKM. The scheme provides straightforward procedures to manage security associations in ad hoc networks. AuthBasedPKM meets a strong set of requirements such as breaking the routing-security interdependence cycle and being progressively robust in a hostile environment.

The entire operation of the scheme eliminates any explicit dependence on the routing infrastructure for certificate delivery; keying material is relayed along the chain without setting up and maintaining a route. As a result, AuthBasedPKM is resistant to frequent route failures and node mobility. The key management scheme overcomes the dynamic network topology by exploiting the routing protocol's inherent ability recover from route failures.

We propose a secure authority-based public key establishment (APKE) protocol that allows an entity to negotiate with an *off-line* trusted authority for a public/private key pair in such a way that the authority does not learn the private key. This protocol is a solid building block for our peer-to-peer key management scheme.

The simplicity of AuthBasedPKM allows for a strong security argument in a widely accepted, adversarial model. The nodes use the underlying key distribution scheme, CertRelay, to distribute only authenticated information on a peer-to-peer basis, which provides provable protection against forgery and undetected modification. The fully distributed scheme preserves the symmetric relationship between the nodes and provides an adversary with no convenient point of attack. We foresee the scheme to be applicable to a wide range of applications that need security associations

to be set up on the network layer, including vehicular networks and peer-to-peer (P2P) applications over heterogeneous networks.

The effectiveness of AuthBasedPKM, its low implementation complexity and ease of integration with existing secure routing protocols were verified through coding and simulating the scheme in ns-2. We show that AuthBasedPKM has negligible impact on the network performance. It was concluded that the message relay mechanism provides an efficient way to manage keying material. The performance of AuthBasedPKM in fact improves as the network topology becomes more dynamic, showing that a dynamic network topology advances security in ad hoc networks.

Part II

Group Key Management in Distributed Communication Systems

Chapter 3

A Survey on Group Key Management for Mobile Ad Hoc Networks

3.1 Introduction

Mobile ad hoc networks allow users to establish communication without any fixed or pre-existing infrastructure. The network therefore has no base stations, access points or remote servers. Nodes that are within each others transmission range communicate directly, while relaying the messages for those too far apart. The mobility of the nodes can lead to 'rapidly changing' (dynamic) network topologies. Nodes do not have any relationships prior to network formation due to the nature of the applications of ad hoc networks [16].

Self-organized ad hoc networks are created solely by the end-users for a common purpose in an unplanned, i.e. *ad hoc* fashion. In contrast to conventional networks the users therefore cannot bootstrap the required security associations with the assistance of a priori shared information on their nodes. This unique property demands distributed collaborative protocols that enable nodes to establish security mechanisms without the assistance of a centralized online Trusted Third Party (TTP).

Many researchers have already proposed peer-to-peer key management schemes that are suitable

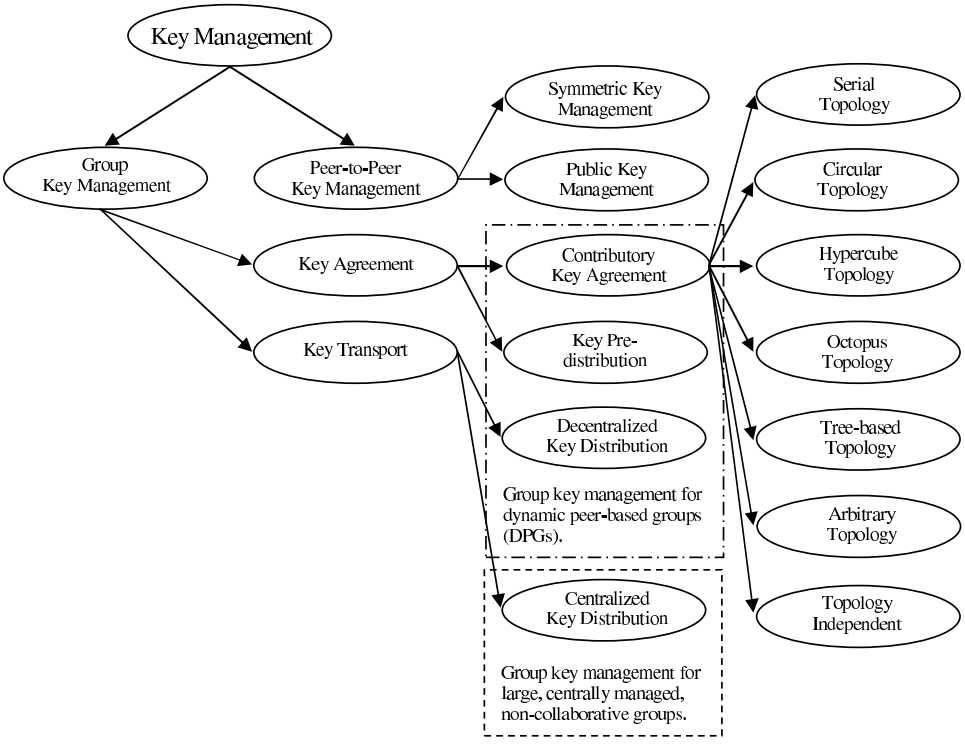

Figure 3.1: Dimensions of group key management.

for self-organized and authority-based ad hoc networks [16]. In contrast, the available literature contains very few group key management schemes that are designed specifically for ad hoc networks. Authors normally adapt group key management schemes for conventional networks to suite the unique characteristics of ad hoc networks.

The military and commercial applications of ad hoc networks incorporate many group-oriented applications [16]. The primary subdivision of group key management schemes emerges from the variety of different application dependent group settings that exist in practice. In Figure 3.1, the two main group key management settings are indicated within the dotted lines, namely:

- Group key management for centrally managed, non-collaborative groups.
- Group key management for *dynamic peer groups* (DPGs).

Large groups, found for example in internet multicast applications, are normally non-collaborative and hard to control on a peer basis [1]. They therefore have a structured hierarchy and exhibit one-to-many broadcast communication patterns [1] [9]. The control structure is maintained by a centralized TTP, chosen prior to network formation.

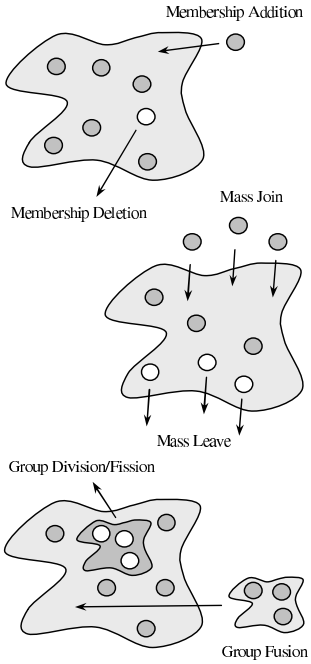

Figure 3.2: Common auxiliary key agreement operations [1].

DPGs tend to be relatively small collaborative groups (with membership in the order of a hundred) where all group participants have a symmetric relationship and must therefore be treated equally [1]. Such systems accordingly have no central point of control. This means that special roles, such as a group controller, are also not fixed prior to group formation, but allocated to any group member during and after group formation. These roles must be assigned based on group policy and must be orthogonal to the key management scheme [1].

DPGs have many-to-many communication patterns and are dynamic in membership, i.e. members join and leave at random. Since a common cryptographic key must be shared between group members at all times, the dynamic membership makes key management protocol design complex as the computational and communication overhead on the network has to be kept to a minimum [9]. Dynamic membership result in group key agreement protocol suites that accommodate initial key agreement (IKA) and auxiliary key agreement (AKA) operations [1]. IKA refers to the key agreement during the first group formation or group genesis, while AKA includes all subsequent key agreement operations. The most common AKA operations are illustrated in Figure 3.2 [1].

Investigations within the available literature have shown that group communication in ad hoc networks will generally occur in the form (or take on the properties) of DPGs. Considering the characteristics and applications of ad hoc networks, as detailed in [16], the dynamic nature of group communications in ad hoc networks as nodes join or leave the network becomes apparent.

Secure key management with the ability to respond to these changes in group membership is at the centre of providing network security for DPGs.

The main objective of this chapter is to analyze the existing group key management protocols for DPGs with respect to their suitability for ad hoc networks.

A fundamental function of key management schemes is the establishment of keying material to facilitate secure communication, that is, it is the first step towards protecting the confidentiality, integrity and availability of information. Key establishment is a process whereby a shared secret (session/symmetric key) becomes available to two or more communication entities, for subsequent cryptographic use [11]. The type of key establishment on which the group key management scheme is constructed, can be used to make a distinction between group key management schemes. As illustrated in Figure 3.1, group key management schemes can either use *key transport* or *key agreement* to establish shared keying material between group members:

- In a group key management scheme based on key transport, a centralized authority or TTP shares a unique secret with each group member. The TTP or designated group controller generates a session key and secretly transfer it to the other group members [1] [11]. The case where a single entity (TTP or key server) generates and distributes keys to the group members is defined as centralized group key distribution [9]. The case where any group member is dynamically chosen as group controller to generate and distribute keys to the other group members is defined as decentralized group key distribution [9].

- In a group key management scheme based on key agreement, the session key is derived collaboratively by all group members as a function of information contributed by, or associated with the members, such that no member can predetermine the resulting shared key [1] [11]. If the individual contribution of each group member remains computationally secret from all other parties (even when any subset of the members collude and use their shares in an attempt to learn the individual secrets of fellow group members) then the scheme is called a contributory key agreement scheme. In [1] contributory key agreement is defined as a protocol that gives a group key $K = H(N_1, \ldots, N_n)$ as output, where $H(\cdot)$ is a collision resistant one-way function and Ni is a secret key share randomly chosen by the i^{th} group member.

  A common offline TTP can also pre-distribute initial keying material to group members prior to network formation. These are referred to as key pre-distribution schemes. Group members can then collaborate on group formation and use the initial keying material to establish a shared group key.

This survey focuses in particular on group key management schemes for DPGs in conventional networks due to their possible relationship to schemes that would be suitable for ad hoc networks. It should however be clear that a group key management scheme designed for DPGs does not necessarily adhere to all the requirements of ad hoc networks. The unique characteristics of ad hoc networks, in particular the dynamic network topology and strong symmetric relationship between nodes, impose some additional constraints on group key management protocols that renders most of the existing schemes for conventional networks impractical.

Investigations within the available publications have shown that the current key establishment protocols for DPGs can be uniquely classified into the following subsets:

1. Key Agreement: Contributory Key Agreement
   (a) Serial topology.
   (b) Circular topology.
   (c) Tree-based topology.
   (d) Arbitrary topology.
   (e) Hypercube topology.
   (f) Octopus topology.
   (g) Star topology.
   (h) Topology independent.
2. Key Agreement: Key Pre-distribution.
3. Key Transport: Decentralized Key Distribution.

The above categorization of group key management schemes for DPGs is also presented in Figure 3.1. Subsets 1a to 1h are based on the topology formed and/or maintained by the protocol participants. The topology is forced upon the group members by the construction method used to obtain a desired group key form, i.e. the structure formed by the group members is determined by the order in which protocol participants perform operations on intermediate keying material or is determined by the order in which the participants generate intermediate keying material. The keying material is operated on or generated in such a way that when these values are combined in a predefined manner, the resulting group key will be in the desired form and will contain a unique random contribution associated with each group member.

As mentioned above, group key management protocols for ad hoc networks share several properties with those suitable for DPGs. Besides the subsets of contributory key management, DPGs can also use group key management schemes that fall under key pre-distribution (Subset 2) as a subset of key agreement and decentralized key distribution (Subset 3) as a subset of key transport (see Figure 3.1).

In the subsequent sections (Section 3.2 to Section 3.11) each of the above subsets (Subset 1a to Subset 1h, Subset 2 and Subset 3) will be introduced by presenting a prominent group key management scheme as representative of the subset. A separate discussion following the introduction of each subset is given. The discussions focus on the subsets' suitability for providing secure and efficient group key management schemes for ad hoc networks. It is hoped that based on this work other researchers will have a view of all the subset variations and be able to propose schemes that takes the dependencies and consequent design constraints into consideration. At the end of the chapter references for further reading are provided such that the reader can apply the analysis approach advocated in this chapter to these group key management schemes.

3.1.1 Organization of Chapter

The chapter is organized as follows: In Section 3.2 the Cliques protocol suite [1] is presented to introduce group key management schemes based on a serial topology. Section 3.3 introduces the Ingemarsson (ING) protocol [66] to represent schemes with circular or ring topologies. Section 3.4 shows how Becker and Wille [67] organize nodes in a hypercube topology to satisfy the lower bound on round complexity for contributory key agreement protocols. In Section 3.5, group key agreement schemes with nodes structured in an octopus topology is considered by introducing the Octopus protocol proposed in [67]. Section 3.6 introduces tree-based schemes by looking at the Tree-based Group Diffie-Hellman (TGDH) protocol suite [9]. Section 3.7 shows how group members in an arbitrary network topology can establish a shared key using the Arbitrary Topology Generalization of Diffie-Hellman (AT-GDH) initial key agreement protocol. In Section 3.8 a star-based topology is considered where a group controller collects and distributes the shares of members in the group key. Section 3.9 presents and extends the initial key agreement protocol by [68] based on a broadcasting system to introduce topology independent schemes. Section 3.10 presents key pre-distribution schemes by discussing the Distributed Key Pre-distribution Scheme (DKPS) [69]. In Section 3.11 the last subset of group key management schemes for DPGs is considered by looking at a modification of the Centralized Key Distribution (CKD) protocol suite [70]. By randomly assigning the responsibility of group controller to any group member, the CKD scheme can be

used as a suitable representative of decentralized key distribution schemes. Section 3.12 analyzes the efficiency of the group key management protocol suites for DPGs that consider both IKA and AKA operations. The chapter is concluded in Section 3.13.

A good understanding of ad hoc networks and suitable peer-to-peer key management protocols are required before considering group key management schemes [16] [13]. Knowledge of 2-party and 3-party key agreement protocols is also a prerequisite [71] [72] [11].

3.2 Contributory Key Agreement: Serial Topology

The first effort to extend the II-party Diffie-Hellman (DH) key exchange protocol [71] to a group setting was proposed by [66] followed by [68]. Later [73] proposed a natural extension of the II-party case to n-parties. These protocols evolved to become the Cliques protocol suite [1], which was the first complete key agreement scheme that incorporated protocols for both IKA operations and the most important AKA operations (see Figure 3.2). The basic idea of the protocols is that the shared key is never transmitted over the network. Instead, a list of partial keys (that can be used by individual members to compute the group secret) is sent. One member of the group, normally called the group controller, is charged with the task of building and distributing this list. The controller is not fixed and has no special security privileges.

The notation that will be used in subsequent text is given below. For mathematical background and explanation of terms, such as "primes", the book by [11] will provide a starting point.

n —Number of protocol participants.

i, j, k, d —Indices of group members $\in [1, n]$.

M_i —*i-th* group member.

M_* —All group members.

p, q —Two large primes, such that $q \mid (p-1)$.

G —Cyclic subgroup of $\mathbb{Z}_p$ with order q.

α —Generator of G.

N_i —Random secret generated by M_i ($N_i \in_R \mathbb{Z}_q$) .

S —Subsets of $\{N_1, \ldots, N_n\}$.

$\prod(S)$ —Product of all elements is the set S.

K_n —Shared group key between n members.

K —Shared group key between n members where n is obvious.

K' —New shared group key after *auxiliary* key agreement.

The CLIQUES protocol suite includes the following protocols:

3.2.1 Cliques Initial Key Agreement: IKA.1

The generic n-party DH protocol [73] is in principle almost identical to the original DH II-party protocol [71]. Members agree *a priori* on G and a generator α of G. For key agreement each member chooses its own secret random number N_i. The generic protocol is based on a *distributive* computation of a subset: $\{\alpha^{\Pi(S)} | \, S \subset \{N_1, N_2, \ldots, N_n\}\}$. From $\{\alpha^{N_1 \cdots N_{i-1} N_{i+1} \cdots N_n}\}$ member M_i can compute the group key

$$K = \{\alpha^{N_1, N_2, \ldots, N_n}\} \tag{3.1}$$

The first initial key agreement protocol, IKA.1, as presented in [1], reduces the number of rounds of the generic case by ordering the group participants in a *serial topology*; in the *upflow* stage each member M_i computes i intermediate values with $i-1$ exponents and one *cardinal* value containing i exponents. For M_i the cardinal value will be: *cardinal* value $= \alpha^{N_1 \cdot N_i}$ which becomes $\alpha^{N_1 \cdot N_{n-1}}$ for M_n. M_n is thus the first member that computes $K_n = \alpha^{N_1 \cdots N_n}$ and in the final stage broadcasts all the intermediate values to the rest of the group which enable the other group members to also compute K.

In summary, IKA.1 allows group members $n \geq 2$ to establish a conference key between them over an insecure channel. The result is a group key K, secure against *passive* adversaries. The protocol participants agree *a priori* on an appropriate prime p and generator α of G.

IKA.1 consists of two stages, upflow and broadcast.

Stage-1. (Upflow): Round i; $i \in [1, n-1]$

$$M_i \longrightarrow M_{i+1} : \{\alpha^{\prod(N_k | k \in [1,i] \wedge k \neq j)} | j \in [1, i], \alpha^{\prod(N_k | k \in [1,i])}\}$$

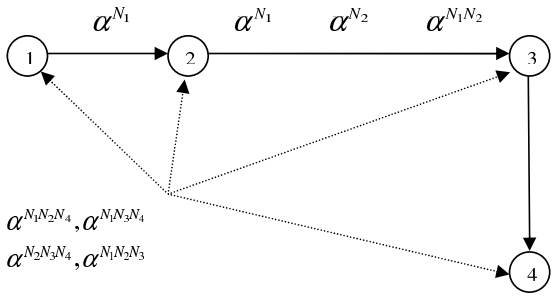

Figure 3.3: Cliques IKA.1 Algorithm - Case of a group with 4 members.

Stage-2. (Broadcast): Round n

$$M_* \longleftarrow M_n : \{\alpha^{\prod(N_k|k\in[1,n]\wedge k\neq j)}|j \in [1,n]\}$$

Each member M_i can then compute the final group key as:

$$K = [\alpha^{\prod(N_k|k\in[1,n]]\wedge k\neq i)}]^{N_i} = \alpha^{\prod_{i=1}^{n} N_i} \tag{3.2}$$

Figure 3.3 provides an example IKA.1 protocol execution between 4 group members.

3.2.2 Cliques Initial Key Agreement: IKA.2

The second initial key agreement protocol presented in [1], IKA.2, reduces the number of exponentiations in comparison to IKA.1 by letting each member, M_i, factor out their exponent in the first *broadcast* stage. IKA.2 has an $upflow$ stage similar to IKA.1. After processing the $upflow$ message, M_{n-1} obtains $\alpha^{\prod(N_k|k\in[1,n-1])}$ and broadcasts this value in the second stage to all protocol participants. On reception, every M_i factors out its own exponent N_i or equivalently divides by N_i and forwards the result to M_n. In the third stage M_n collects all inputs from $M_1, \cdots, M_{n-1}$ and raises every one of the $n-1$ inputs to its secret value N_n. The last member in the chain (M_n) then finally broadcasts the resulting intermediate values to the rest of the group. Since all members ultimately have a value of the form $\alpha^{\prod(N_k|k\in[1,n]\wedge k\neq i)}$, they can all generate the group key K.

IKA.2 consists of four stages: upflow, broadcast, response and a final broadcast stage.

Stage-1. (Upflow): Round i; $i \in [1, n-2]$

$$M_i \longrightarrow M_{i+1} : \{\alpha^{\prod(N_k|k\in[1,i])}\}$$

Stage-2. (Broadcast): Round $n-1$

$$M_* \longleftarrow M_{n-1} : \{\alpha^{\prod(N_k|k\in[1,n-1])}\}$$

Stage-3. (Response): Round n

$$M_i \longrightarrow M_n : \{\frac{\alpha^{\prod(N_k|k\in[1,n-1])}}{N_i}\}$$

Stage-4. (Broadcast): Round $n+1$

$$M_* \longleftarrow M_n : \{\frac{\alpha^{\prod(N_k|k\in[1,n])}}{N_j}|j \in [1,n]\}$$

3.2.3 Cliques Auxiliary Key Agreement: Membership Addition

All AKA operations take advantage of the keying material collected during the *upflow* stage of the latest IKA or AKA protocol execution. Any member who cached the keying material of the most recent broadcast round can facilitate an AKA operation by taking on the responsibility of the group controller (M_C). Due to the similarity between the CLIQUES AKA protocols only the *membership addition*, *merge* and partition protocols will be considered. The reader is referred to [1] for details on the other AKA operation protocols of CLIQUES.

The membership addition protocol adds a single member to the existing group. The group members, including the new member, establish a new conference key K' between them that is secure against *passive* adversaries. In the membership addition protocol, M_C extends the last round of the prior IKA or AKA protocol's *upflow* stage by one more round. The new member raises the keying material to its own new random secret before broadcasting the value to all members in the second round. Note that the group controller replaces its own secret, N_C, in the *upflow* message by N'_C to prevent new members from obtaining old group keys.

The CLIQUES membership addition protocol requires the following two stages:

Stage-1. (Upflow): Round 1 ($N_C \longleftarrow N_C N'_C$)

$$M_C \longrightarrow M_{n+1} : \{\alpha^{N'_C \prod(N_k | k \in [1,n] \wedge k \neq j)} | j \in [1,n], \alpha^{N'_C \prod(N_k | k \in [1,n])}\}$$

Stage-2. (Broadcast): Round 2

$$M_* \longleftarrow M_{n+1} : \{\alpha^{N'_C \prod(N_k | k \in [1,n+1] \wedge k \neq j)} | j \in [1, n+1]\}$$

3.2.4 Cliques Auxiliary Key Agreement: Merge Protocol - GDH IKA.3

The merge protocol is designed to accommodate network merge, i.e. k members are added to a group of n members. The protocol, as described in [70] [74], works as follows. As in the case of membership addition protocol, when a merge event occurs, the current group controller M_C (which can be any member of the existing group) generates a new key token by refreshing its contribution to the group key and then passes the token to one of the new members. When the new member receives this token, it adds its own contribution and passes the token to the next new member. This continues till the token reaches the last new member. This member (the last one in the new group) becomes the new group controller. The controller broadcast the token to the group without adding its contribution. Upon receiving the broadcast token, each group member (old and new) factors out its contribution and unicasts the result (called a factor-out token) to the new group controller. The new controller collects all factor-out tokens, adds its own contribution to each of them, building the list of partial keys and broadcasts it to the group. Every member can then obtain the group key by factoring in its contribution.

The Merge-Protocol requires the following steps:

Stage-1. M_C generates a new exponent $N'_C (\in Z_q)$, and sends to M_{n+1} the message:

$$M_C \longrightarrow M_{n+1} : \{\alpha^{N'_C \prod(N_i | i \in [1,n] \wedge i \neq C)}\}$$

Stage- $j+1$ for $j \in [1, k-1]$. New merging member M_{n+j} generates an exponent N_{n+j} and forwards to member M_{n+j+1} the message

$$M_{n+j} \longrightarrow M_{n+j+1} : \{\alpha^{N'_C \prod(N_i | i \in [1,n+j])} | j \in [1, k-1]\}$$

Stage- $k+1$. Upon receipt of the accumulated value, M_{n+k}, who is now the new group controller,

broadcast it to the entire group.

Stage- $k+2$. Upon receipt of the broadcast, every member M_i, $\forall i \in [1, n+k-1]$ sends back to M_{n+k} the message:

$$M_{n+k} \longleftarrow M_* : \{\alpha^{N'_C \prod(N_j | j \in [1,n+k-1] \wedge j \neq i \neq C)} | i \in [1, n+k-1]\}$$

Stage- $k+3$. After collecting all responses M_{n+k} generates a new exponent N_{n+k} and broadcasts the set S to all the members of the group:

$$M_* \longleftarrow M_{n+k} : S = \{\alpha^{N'_C \prod(N_j | j \in [1,n+k] \wedge j \neq i \neq C)} | \forall i \in [1, n+k-1]\}$$

Stage- $k+4$. Upon receipt of the broadcast, every member M_i, $\forall i \in [1, n+k]$ computes the key:

$$K = (\alpha^{N'_C \prod(N_j | j \in [1,n+k] \wedge j \neq i \neq C)})^{N_i} = \alpha^{N'_C \prod(N_j | j \in [1,n+k] \wedge j \neq C)} = \alpha^{N_1 \ldots N'_C \ldots N_{n+k}} \mathrm{mod} p$$

3.2.5 Cliques Auxiliary Key Agreement: Partition Protocol

GDH-partition protocol considers the case when a number of members leave a group [70] [74]. In case of partition, L members are leaving a group of size n. The group controller who is at all time the most recent remaining group member, removes the corresponding partial products of the members leaving from the list of partial keys. It refreshes each partial key in the list and broadcasts the list to the group. Each remaining member can then compute the shared key. The protocol runs as follows:

Step-1. The group controller M_C generates a new exponent N'_d , broadcast the set S:

$$M_* \longleftarrow M_C : S = \{\alpha^{N'_C \prod(N_j | j \notin L \wedge j \neq i \neq C)} | \forall i \notin L\}$$

Stage-2. Upon receipt of S, every remaining member M_i, $\forall i \notin L$ computes the key K:

$$K = (\alpha^{N'_C \prod(N_j | j \notin L \wedge j \neq i \neq C)})^{N_i} = \alpha^{N'_C \prod(N_j | j \notin L)} = \alpha^{N_1 \ldots N'_C}$$

3.2.6 Discussion of the Cliques Protocol Suite

The CLIQUES protocol suite was designed for DPGs in conventional networks. CLIQUES therefore does not allow for some of the unique characteristics of ad hoc networks. CLIQUES depends on a serial execution of computations in the *upflow* stages of its IKA protocols and thus requires nodes to form a serial topology. Considering the dynamic network topology of ad hoc networks, caused by node mobility, this is a critical inefficiency. The serialized topology of nodes in the *upflow* stages of the IKA protocols is thus more suited to static networks.

The CLIQUES protocol suite is only secure against *passive* adversaries. Member authentication mechanisms, which are required to thwart *active* adversaries are not specified in the CLIQUES protocol suite. It should be noted that incorporating such mechanisms will result in an increase in the communication and computational cost of CLIQUES [75].

CLIQUES fails to distribute the burden of group key management equally between all members, this in itself is sufficient justification to declare CLIQUES inappropriate for ad hoc networks. The fact that CLIQUES unevenly distributes the computational and communication overhead between all group members is experimentally substantiated in [76]. For example, the first node in the serial *upflow* stage of IKA.1 (Section 3.2.1) performs only a single exponentiation where the last node P_n performs n exponentiations. In the case of conventional networks, which in general have sufficient communication, memory and energy resources, an uneven distribution of the overhead should cause no concern. In ad hoc networks these resources are limited. An uneven distribution of computational overhead also makes the CLIQUES IKA protocols vulnerable to a *selfishness attack* [77] by *legitimate* group members. The highest-indexed group member P_n has to broadcast all the intermediate values to the other group members in the final round of the IKA protocols. P_n thus plays a *special* role and therefore provides adversaries with a single point of attack.

In the AKA protocols, the disadvantages of CLIQUES are also apparent. All the AKA protocols rely on the assumption that at least one arbitrary group member caches all the keying material broadcast during the last round of the IKA operation or previous AKA operation. This group member *voluntarily* takes over the functionality of group controller in the case of a subsequent AKA operation. Considering the resources required to cache the keying material and to perform the role of group controller, it should be clear that the CLIQUES protocol suite violates the symmetric relationship between the group members and is therefore subject to *selfishness*. In order to guarantee that a legitimate group member takes on the responsibility of group controller, this role will have to be assigned based on group policy. In the AKA operations for membership ad-

dition and mass join, the highest-indexed new member also plays the special role of broadcasting the intermediate values to the other group members. The highest-indexed member (and group controller) in the AKA protocols thus present a single point of vulnerability.

Lately the authenticated CLIQUES protocols have been shown to suffer from generic insecurities [78].

Amir et al. [79] build on the Group Diffie-Hellman proposed by [1] to construct a robust contributory key agreement protocol resilient to a finite sequence of auxiliary key agreement operations. More specifically, the robust algorithm optimized the group change protocols (e.g. join, leave, merge etc.) and model these auxiliary operations by a state machine. The optimized algorithm is implemented using the services of the Secure Spread Library [80].

3.3 Contributory Key Agreement: Circular Topology

In literature the Ingemarsson protocol, generally referred to as ING, belongs to the family of group key agreement protocols proposed in [66]. It requires a synchronous startup and executes in $(n-1)$ rounds, where n is the total number of protocol participants. The members must be arranged in a ring or *circular topology*. In a given round every protocol participant raises the previously received intermediate key value to the power of its own exponent (random secret) and forwards the result to the next participant in the ring. Consequently ING also falls under the family of protocols that are a *natural* extension of the II-party DH protocol [1]. After $(n-1)$ successfully completed rounds all n participants in the ring share a group key K.

3.3.1 Ingemarsson Initial Key Agreement: ING

The ING protocol allows $n \geq 2$ group members, ordered in a circular topology, to establish a group key K as follows:

Round k; $k \in [1, n-1]$.

$$M_i \longrightarrow M_{(i+1)\ mod\ n} : \{\alpha^{\Pi(N_j | j \in [(i-k)\ mod\ n, i])}\}$$

3.3.2 Discussion of the Ingemarsson Protocol

The Ingemarsson protocol [66] is one of the earliest attempts to provide contributory key agreement by extending the DH II-party case to group settings for teleconferencing. The ING protocol requires participants to set up a logical ring or circular formation. The circular topology requirement may prohibit its use when one considers the characteristics of ad hoc networks. As a result of forming the ring structure, the ING protocol compels group members to maintain track of the availability of their ordered neighbors at all times. It is known that ad hoc networks are subject to error-prone wireless connectively and numerous link attacks as they may operate in hostile networking environments [81] [3]. Another two challenges in providing key management for ad hoc networks are due to node mobility and frequent network partitioning. As a consequence to these mobile ad hoc network characteristics the ring structure is difficult to maintain.

The ING protocol is considered to be inefficient for the following reasons:

- At startup, synchrony is required between all members forming the group.
- A total of $n-1$ simple rounds are required to establish the conference key, K_n.
- The symmetrical nature of the protocol makes dynamic membership support a costly operation.
- The n exponentiation required by each joining member is not feasible for computationally constrained devices.

The ING protocol also does not scale very well since the total exponentiation is $O(n^2)$. The computational cost thus grows exponentially with group membership.

The ING protocol is insecure allowing a *passive* adversary eavesdropping on the information exchanged between users to compute the conference key K [68]. The proposal by Ingemarsson *et al.* does not provide any mechanisms for authentication or AKA operations and this could be regarded as a shortcoming.

It may thus be concluded that the ING protocol is not suitable for ad hoc networks, mainly due to the ING protocol's inefficiency, lack of support for AKA operations, inherited insecurity and underlying circular topology.

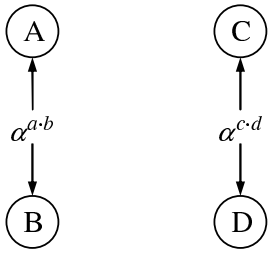

Figure 3.4: Round 1 of 2^2 Hypercube protocol

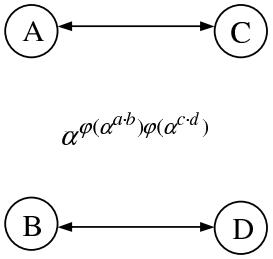

Figure 3.5: Round 2 of 2^2 Hypercube protocol

3.4 Contributory Key Agreement: Hypercube Topology

The Hypercube protocol by [67] achieves the lower efficiency bound with respect to number of rounds (as defined in [67]). The protocol accomplishes this by grouping member into groups of four. The four parties, A, B, C and D in each grouping can establish a secret key between them using four DH exchanges. As shown in Figure 3.4 [67], parties A and B exchange keys using the II-party DH protocol and concurrently C and D perform the same action. In the second round shown in Figure 3.5 [67], parties A and C exchange keys using the II-party DH protocol and concurrently B and D perform the same action. The resultant IV-party DH key is in the following form: $K = \alpha^{\alpha^{a \cdot b} \alpha^{c \cdot d}}$

Assume there are 2^3 participants compared to 2^2, i.e. two groupings of four. The 8 participants are arranged as the vertices in a 3-dimensional *hypercube topology*. In the 1^{st} and 2^{nd} round, each group of four participants can establish a IV-party DH key value as explained above. In the 3^{rd} and final round, each participant on a square face performs exchanges with its peer on the opposite square face using its IV-party secret key from the first two rounds as the exponent. This will result in a shared key, K_8, between the participants. This process will continue for another round for 2^4 participants and another two rounds for 2^5 etc. Thus in round i, each of the protocol participants performs a II-party DH with its peer on the i^{th} dimension of the hypercube using the key of round $i-1$ as its secret exponent. After d rounds, 2^d participants will have the same secret key K_{2^d}.

For example the shared key generated by the 8 members can be represented as:

$$K = \alpha^{\{\alpha^{\{\alpha^{\{r_1 \cdot r_2\}} \alpha^{\{r_3 \cdot r_4\}}\}} \alpha^{\{\alpha^{\{r_5 \cdot r_6\}} \alpha^{\{r_7 \cdot r_8\}}\}}\}} \tag{3.3}$$

Before continuing, the additional notation required in the following text is defined.

d —Cube dimension.

$GF(2^d)$ —d-Dimensional vector space.

φ —Bijection: $G \longrightarrow \mathbb{Z}_q$.

$\overrightarrow{v}$ —Vector representing each participant in $GF(2^d)$.

$r_{\overrightarrow{v}}, r_i$ —Random secret generated by participant $\overrightarrow{v}$ or member M_i of n.

3.4.1 Hypercube Initial Key Agreement

The Hypercube protocol allows 2^d participants ordered in a hypercube topology to derive a conference key, K, secure against *passive* adversaries. All participants agree on an appropriate prime p and generator α of G. Members select their secrets r_i randomly from G. Parties are identified with linearly independent vectors spanning the d-dimensional vector space $GF(2^d)$ with basis $\overrightarrow{b}_1, \overrightarrow{b}_2, \ldots, \overrightarrow{b}_d$.

Hypercube protocol executes in d rounds as follows:

Round-1.

- **(a)** $\overrightarrow{v}$ generates secret exponent $r_{\overrightarrow{v}}$.
- **(b)** $\overrightarrow{v}$ performs II-party DH exchange with $\overrightarrow{v} + \overrightarrow{b_1}$ using $r_{\overrightarrow{v}}$ as its random secret exponent.

Round-i ($1 < i \leq d$).

- **(a)** $\overrightarrow{v}$ performs II-party DH exchange with $\overrightarrow{v} + \overrightarrow{b_i}$ using the DH key generated in round $i-1$ as the secret exponent.

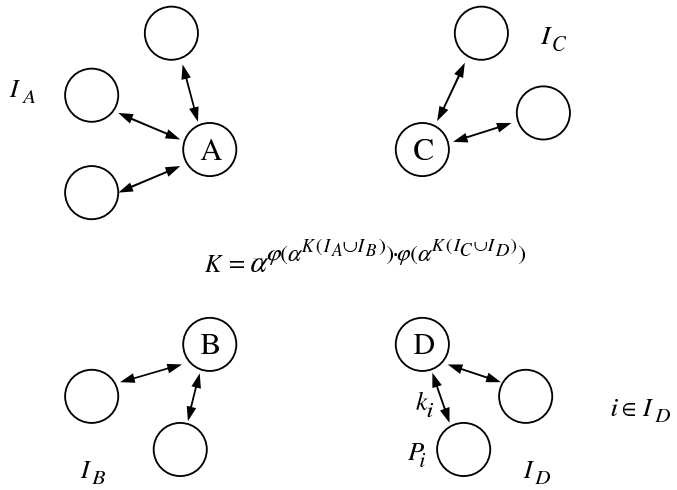

Figure 3.6: Network topology of Octopus protocol

3.4.2 Discussion of the Hypercube Protocol

The Hypercube protocol is only feasible for an even number of group members which may be regarded as a limiting feature. Becker and Wille [67] use the Hypercube protocol as a foundation for the *Octopus protocol* reviewed in Section 3.5. The reader is referred to the discussion on the Octopus Protocol in Section 3.5.2 for further comments on the Hypercube protocol's suitability for ad hoc networks, due to the close relationship between the Hypercube and Octopus protocols.

3.5 Contributory Key Agreement: Octopus Topology

The Octopus protocol presented in [67] addresses the scenario when the number of participants that form a group is not of the power 2. The Octopus protocol eliminates the limitation of the Hypercube protocol by allowing an arbitrary number of protocol participants to establish a group key as follows: In the Octopus protocol, four participants form the *central* control-core of the group. As illustrated in Figure 3.6 [67], the rest of the participants attach to one of the four central controllers to form an *octopus topology*. These attachments are also referred to as 'tentacles' or pair wise disjoint groups. All the members perform a II-party DH exchange with their respective central controller. The four central controllers then perform a IV-party DH exchange as defined in Section 3.4, using the product of the tentacle keys as secret exponents.

The following additional notation is defined as required in the subsequent text.

A, B, C, D —Controlling group members.

X —Set of central controlling nodes; $X \in \{A, B, C, D\}$.

I_X —Pair wise disjointed subgroup connected to X.

P_i —Non-controlling participants: $\{P_i | i \in I_X\}$.

r_{P_i}, r_X —Random secret generated by P_i and X respectively.

φ —Bijection: $G \longrightarrow \mathbb{Z}_q$.

Assume that parties $P_{n-3}, P_{n-2}, P_{n-1}$ and P_n form the set X. The rest of the participants $P_1, \ldots, P_{n-4}$ are connected to one of $X \in \{A, B, C, D\}$.

The Octopus protocol comprises the following three steps:

Step-1.

(a) All X generate secret exponent r_X and all P_i for $i \in [1, n-4]$ generate secret exponent r_{P_i}.

(b) For all X and $i \in I_X$, X and P_i performs a II-party DH key exchange.

Step-2.

(a) The controlling nodes perform a IV-party DH exchange or 2^2 Hypercube protocol using the secret exponents $a, b, c, d = K(I_X)$, where $K(J) = \prod_{i \in J} \varphi(k_i)$ for $J \subseteq \{1, \ldots, n-4\}$.

Step-3.

(a) Controller node A sends to all P_i, $i \in I_A$, the following values: $\alpha^{K(\frac{i}{I_B \cup I_A})}$ and $\alpha^{\varphi(\alpha^{K(I_C \cup I_D)})}$. The remaining controller nodes B, C and D also send their two values to P_i, $i \in \{I_B, I_C, I_D\}$ respectively.

The resulting group key is computed as:

$$K = \alpha^{\{\varphi(\alpha^{K(I_A \cup I_B)})\varphi(\alpha^{K(I_C \cup I_D)})\}} \tag{3.4}$$

3.5.1 2^d-Octopus Protocol

An alteration to the Octopus protocol, called the 2^d-Octopus protocol is also presented in [67]. If n is the number of participants and $(2^d < n < 2^{d+1})$, the first 2^d participants perform the

functionality of the central controllers in contrast to only four in the Octopus protocol. The remaining nodes attach to one of the 2^d controller nodes. The 2^d scheme follows the same algorithm as the Octopus protocol (Section 3.5) to establish a group key known to all members.

3.5.2 Discussion of the Octopus - and 2^d-Octopus Protocols

Becker and Wille [67] propose the Octopus protocol and 2^d-Octopus protocol as an example of a DH based group key agreement protocol that satisfies the lower bound on efficiency for message exchanges and round complexity respectively. In contrast to the 2^d-Hypercube protocol (Section 3.4), the Octopus and 2^d-Octopus protocols can accommodate an uneven number of group members. In the case of $n \neq 2^d$ the minimal number of simple rounds is exceeded by one ($d = \log_2 n + 1$) and in the case where $n = 2^d$ the minimal number of simple rounds ($d = \log_2 n$) is equaled.

In [67] the authors do not consider AKA operations. Investigations on the 2^d-Hypercube, Octopus and 2^d-Octopus protocols show that membership additions may be performed efficiently, but membership exclusions fail completely. [1] point out that splitting the group on the d^{th} dimension into two halves seems to be the only possible exclusion procedure. Efficiently performing AKA operations in ad hoc networks is essential since these operations are guaranteed to occur more frequently than in conventional network settings.

The hypercube and octopus topologies make the 2^d-Hypercube, Octopus and 2^d-Octopus protocols unsuitable for ad hoc networks for reasons similar to those making the Cliques protocol suite's *serial topology* (Section 3.2.6) and the ING protocol's *circular topology* (Section 3.3.2) unsuitable for ad hoc networks.

The protocols are also not *fully* distributed therefore violating the symmetric relationships between the members in ad hoc networks. The members forming the control-core are a central point of vulnerability, which is not ideal in ad hoc networks.

[82] propose improvements on the 2^d-Octopus protocol. Their protocol use the *password authenticated* II-party DH key agreement protocol (as a specific case of the generic *encrypted key exchange* protocol [83]) to defend against *active* adversaries. In [82], the II-party DH key exchanges used in the 2^d-Octopus protocol are simply replaced by the *password authenticated* II-party DH key agreement protocol given above.

The password authenticated group key agreement protocol presented by [82] considers a collaborative networking scenario where small groups form an ad hoc network. The key agreement

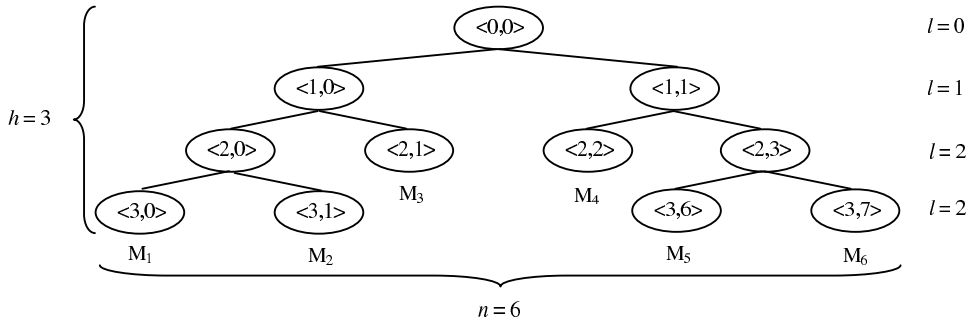

Figure 3.7: *TGDH* key tree model example

scenario in itself is fairly restrictive and the requirement for distribution of a password between members *a priori* limits the practicality of the protocol. Although the proposal incorporates a *weak* authentication mechanism into the 2^d-Octopus protocol it fails to eliminate the protocol's other shortcomings as given above.

3.6 Contributory Key Agreement: Tree-based Topology

In tree-based group key management schemes, keys are organized into a tree hierarchy, based on different construction strategies. The motivation behind employing a *tree-based topology* is to reduce the rekeying cost by localizing the effects of AKA operations (Figure 3.2). This provides improved scalability especially for secure communications in large dynamically changing groups.

The Tree-based Group Diffie-Hellman (*TGDH*) protocol suite [9] provides a solution with a full array of IKA and AKA protocols, which are very similar to the CLIQUES protocols [1]. The only major difference is the organization of nodes during the key agreement process. *TGDH* improves on the efficiency of CLIQUES, with respect to AKA operations, by reordering group members in a binary tree structure. The effect of group events (members joining, leaving etc.) (as mentioned above) becomes localized and therefore reduces the effect of the group dynamics on the protocol efficiency. Kim et al. [9] do not explicitly specify any IKA protocol and therefore *TGDH* cannot be compared to any other IKA group protocols.

In Figure 3.7 [9] a *TGDH* key tree example is given to illustrate the *TGDH* key tree model. The example is used by [9] to explain the notation of *TGDH* key trees and the fundamental operation of the protocol.

The root of the *binary* tree is located at *level* 0 and the depth of the tree may extend to *level* h. In

the binary tree, nodes may be either be a *leaf* or a *parent* of two other nodes. A node key is derived from the contribution of the two children via a II-party DH key agreement protocol execution. In the tree model, *leaf* nodes are denoted by $< l, v >$, where $0 \leq v \leq 2^l - 1$, since each *level* l can host at most 2^l nodes. Every node, $< l, v >$ in the tree is also associated with a key $K_{<l,v>}$ and a blinded key $BK_{<l,v>} = f(K_{<l,v>})$ where $f()$ is a modular exponentiation in a prime order group (i.e. $f(k) = \alpha^k \ mod \ p$). Group members M_i ($i \in [1, n]$) are hosted by the *leaf* nodes, $< l, v >$ and know its random session key $K_{<l,v>}$. The member M_i located at node $< l, v >$ also knows every key along the *key path* from $< l, v >$ to the root node $< 0, 0 >$. The key path is denoted as KEY_i^*.

Before continuing it would be beneficial to take note of the following additional notation.

h —Height of binary tree.

$< l, v >$ —v^{th} node at level l in binary tree.

T_i —M_i's view of the key tree.

$\widehat{T}_i$ —M_i's modified view of the key tree.

$H(\cdot)$ —Collision free one-way hash function.

r_i —Random secret chosen by member M_i.

In Figure 3.7 [9], member M_2 owns the tree T_2 and knows every key $\{K_{<3,1>}, K_{<2,0>}, K_{<1,0>}, K_{<0,0>}\}$ in the key path, $KEY_2^* = \{< 3, 1 >, < 2, 0 >, < 1, 0 >, < 0, 0 >\}$ and every blinded key $BK_2^* = \{BK_{<0,0>}, BK_{<1,0>}, \ldots, BK_{<3,7>}\}$.

Every key is computed recursively as: $K_{<l,v>} = f(K_{<l+1,2v>} K_{<l+1,2v+1>})$. Computing $K_{<l,v>}$ thus requires the knowledge of the keys of the two children nodes and the blinded key ($BK_{<l,v>}$) of the other child. $K_{<0,0>}$ is the shared secret by all M_i. The final group key is derived from $K_{<0,0>}$ as $K_{GROUP} = H(K_{<0,0>})$.

For example M_2 in Figure 3.7 can compute the group key $K_{<0,0>}$ as:

$$K_{<0,0>} = \alpha^{(\alpha^{r_3(\alpha^{r_1 r_2})})(\alpha^{r_4(\alpha^{r_5 r_6})})} \tag{3.5}$$

To simplify the protocol description, the authors of [9] introduced the term *co-path*, denoted as CO_i^*. CO_i^* is the set of siblings to each node in the key path on tree T_i as seen by M_i, i.e. every

member M_i at leaf node $< l, v >$ can derive the group secret $K_{<0,0>}$ from all blinded keys on the CO_i^* and its own random session key $K_{<l,v>}$.

TGDH, similar to CLIQUES [1], provides protocols in support of all the common AKA operations. These include the join, leave, merge, partition and key refresh operations. *TGDH* requires certain members to take on a *special* role during the AKA operations. These members, called *sponsors*, have the responsibility of computing blinded keys and broadcasting these to the other group members. Two of these operations, join and leave will be given further consideration.

3.6.1 *TGDH* Auxiliary Key Agreement: Membership Addition

The new member joining the group M_{n+1}, initiates the protocol by broadcasting a join request message that contains its own blinded key $BK_{<0,0>}$. When the current group members receive this message, they determine a new *insertion node*. The insertion occurs at the shallowest rightmost node that will not increase the tree depth h. The *sponsor* is defined as the rightmost leaf node in the subtree rooted at the insertion node. The sponsor generates a new intermediate node and a new member node, and promotes the new intermediate node to the parent of its node and the new member's node. After updating the tree, only the sponsor can compute the group key since it is the sibling of the joining node and knows all the necessary blinded keys. After computing the group key, the sponsor broadcasts the new tree $\widehat{T}_i$ containing all blinded keys. All other members update their tree using the contained information, and compute the new group key K'.

The *TGDH* membership addition protocol executes in three steps:

Step-1. New member M_{n+1} broadcasts a request to join the group.

$$M_{n+1} \longrightarrow M_* : BK_{<0,0>} = \alpha^{r_n+1}$$

Step-2. Binary tree update procedure.

(a) Each M_i does the following:

(i) Updates its key tree by adding a new member node and new intermediate node.

(ii) Removes all keys and blinded keys from the leaf node, related to the sponsor, up to the root node.

(b) The sponsor of the intermediate node M_s additionally:

(i) Generates a new share and computes all $[K; BK]$ pairs on the KEY^*.

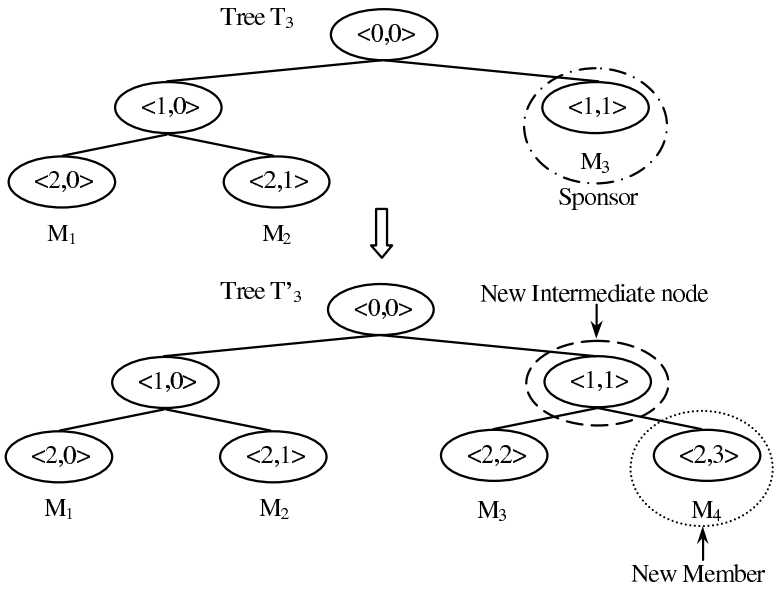

Figure 3.8: *TGDH* update: join example

(ii) Broadcasts updated tree $\widehat{T_s}$ including only blinded keys.

$$M_* \longleftarrow M_s : \widehat{T_s}(BK_s^*)$$

Step-3. All M_i compute K' using $\widehat{T_s}$.

In Figure 3.8 [9], an example is shown of member M_4 joining a group, where M_3 is the sponsor, M_s and performs the following actions:

1. Renames node $< 1, 1 >$ to $< 2, 2 >$.
2. Generates a new intermediate node $< 1, 1 >$ and a new member node $< 2, 3 >$.
3. Promotes $< 1, 1 >$ as the parent node of $< 2, 2 >$ and $< 2, 3 >$.

All members know $BK_{<2,3>}$ and $BK_{<1,0>}$. M_3 can compute the new group key $K_{GROUP} = H(K_{<0,0>})$. Note that $K_{<0,0>}$ is never used for real communication. Members use a strong one-way hash function to compute K_{GROUP}, which improves the randomness of the group key. Every other member also performs step 1 and 2, but cannot compute the group key in the first round. Upon receiving the broadcast blinded keys from M_s, every member can compute the new group key.

3.6.2 *TGDH* Auxiliary Key Agreement: Membership Exclusion

Assume member M_d leaves the group. In this case, the sponsor M_s is the sibling node of M_d. If the sibling is not a leaf node, the sponsor is the right-most leaf node of the subtree which has the sibling node as root of the subtree. In the leave protocol, every member updates its key tree by deleting the node of M_d and its parent node. The sponsor picks a new secret share, computes all keys on its key path up to the root and broadcasts the new blinded keys of its key path to the group. This information allows all members to recompute the group key.

The *TGDH* membership exclusion protocol executes in two steps:

Step-1. Binary tree update procedure.

(a) Each M_i does the following:

(i) Updates key tree by removing the leaving member node and relevant parent node.

(ii) Removes all keys and blinded keys from the leaf node, related to the sponsor, to the root node.

(b) The sponsor M_s additionally:

(i) Generates new share and computes all $[K; BK]$ pairs on the KEY^*.

(ii) Broadcasts updated tree $\widehat{T_s}$ including only blinded keys.

$$M_s \longrightarrow M_* : \widehat{T_s}(BK_s^*)$$

Step-2. All M_i compute K' using $\widehat{T_s}$.

In Fig. 3.9 [9], an example is shown of member M_3 leaving a group, with M_5 as the sponsor. Every remaining member deletes $< 1, 1 >$ and $< 2, 2 >$, i.e. updates the key tree. The sponsor ,M_s, performs the following additional actions:

1. Picks a new share $K_{<2,3>}$ and recomputes $K_{<1,1>}$, $K_{<0,0>}$, $BK_{<2,3>}$ and $BK_{<1,1>}$.
2. Broadcasts the updated tree $\widehat{T_5}$ with BK_5^* included.

After receiving the broadcast message from M_s, every member can compute the new group key. The excluded member M_3 does not hold a share in the group key and therefore cannot compute the new group key K'.

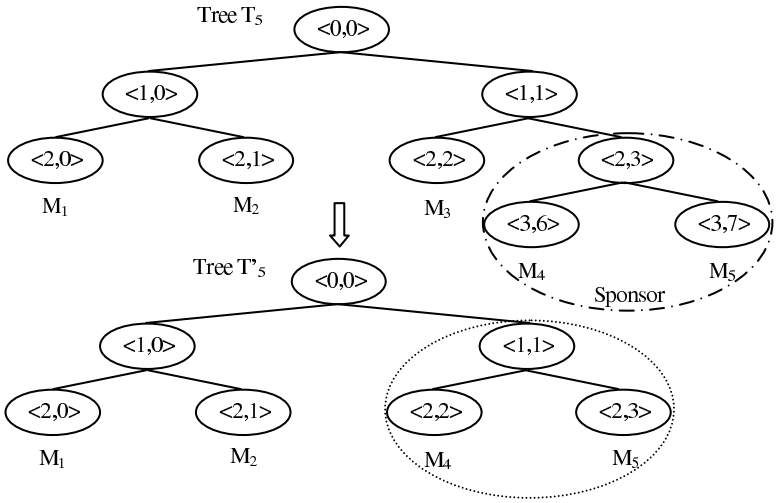

Figure 3.9: *TGDH* update: leave example

3.6.3 Discussion of the *TGDH* Protocol Suite

As in the case of CLIQUES (Section 3.2), the *TGDH* protocols do not consider the unique characteristics of ad hoc networks. The mobility of the nodes results in a dynamic network topology, where dynamic in this case should not be confused with dynamic group membership events. The dynamic nature of the network topology has the consequence that protocols cannot rely on any form of order or structure that is dependent on the network topology.

The burden of group key management in *TGDH* is not equally distributed to all group members. As explained in Section 3.1, groups in ad hoc networks are mostly DPGs where all members have a strong symmetric relationship and should be treated equivalently. As *TGDH* fails to distribute the communication and computational overhead fairly between group members, it may be subject to *selfishness* [77]. Steiner *et al.* also point out that the special roles performed by members should be left to group policy and should be orthogonal to the key management scheme [1]. In *TGDH*, the majority of the work during AKA operations is performed by the sponsor nodes. The members thus rely on the sponsor to facilitate AKA operations . Assuming the sponsor to be available and always able to perform the associated tasks correctly is impractical in ad hoc networks and introduces a single point of failure. Similar to the highest-indexed member in CLIQUES (Section 3.2), M_s also plays a *special* role and therefore provides adversaries with a single point of attack.

In [84] [85] tree based key agreement is revised within a strong adversarial model. The adversary (A) is considered as a Probabilistic Polynomial-Time (PPT) algorithm with complete network control. This gives A the ability to participate on behalf of corrupted group members. The paper

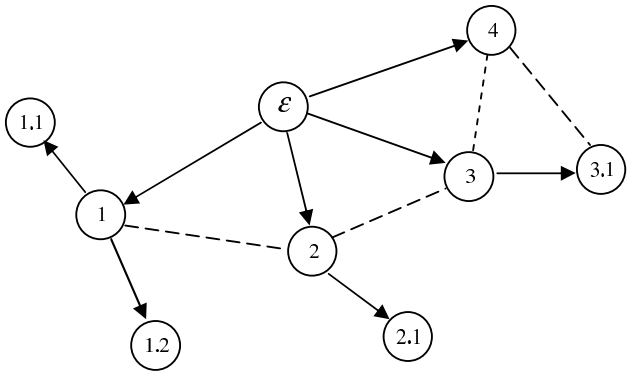

Figure 3.10: Constructing a spanning tree

and the extended version propose a protocol constructed on the main mechanisms of the TGDH protocol and prove the revised protocol secure within the adversarial model.

Desmedt et al. [86] recall the tree-based BD protocol (BD-II). Burmester and Desmedt [87] in turn extend and generalize the conference scheme presented in [68]. The BD-II protocol is improved from $O(n)$ computational complexity per user to $O(logn)$. Desmedt et al. [86] further adjusts the compiler of Katz and Yung [88] to preserve the improvement in computational complexity under a transformation from an unauthenticated to an authenticated group key agreement protocol.

3.7 Contributory Key Agreement: Arbitrary Topology

Hietalahti [89] recognizes the unsuitability of *topology dependent* group key agreement protocols for ad hoc networks. The protocol in [89] adapts the $TGDH$ IKA algorithm [9] to accommodate an *arbitrary* network topology and therefore be more suitable for ad hoc networks. The protocol is called Arbitrary Topology Generalization of Diffie-Hellman (AT-GDH). The protocol, in contrast to $TGDH$, constructs an arbitrary spanning tree dependent on the physical location of the members in the network at the time of group formation.

The initial state of the protocol assumes that the group members know their neighbors and have bidirectional communication. The group formation *initiator* sends a message to each of its neighbors. The initiator becomes the root of the tree and the neighbors its children. The children send a similar message to their neighbors, excluding the parent. The nodes acknowledging the message become children. The process repeats until all group members have received a message. A leaf is defined as a node that does not receive any acknowledgements. Fig. 3.10 [89] illustrates the construction of such an arbitrary spanning with initiator or root ϵ.

Once the tree is constructed, the protocol gathers contributions towards the group key in the first phase. All the leaves of the spanning tree, i.e. nodes with no children, generate a secret value, compute its share and send their share to the parent nodes. The process continues until the root has collected all the contributions from all children. The root then makes the final contribution. In the second phase the root starts to distribute the keying material down the tree towards all children that use the information to compute the group key.

The following additional notation is defined:

ϵ —Protocol initiator (root node of spanning tree).

k_x, e_x —Random secret generated by node x.

c_x —Number of children connected to node x.

$m_{x.i}$ —Message which member x sends to its child i.

As illustrated in Fig. 3.10, nodes have a *universal address* in a rooted tree sequence. The address of the root ϵ is an empty sequence. The children of the root have addresses $1, 2, 3, \ldots$ respectively. When a node x in *level* $l \geq 1$ has children their addresses are given as $x.1, x.2, x.3, \ldots$.

A graph is a pair $G = (V, E)$ where V is a finite non-empty set and $E \subseteq \{< x, y > | x \in V, y \in V, x \neq y\}$. The elements of V are called *nodes* and the elements of E are called *edges*. An undirected (minimally connected) graph is a tree if it is connected and does not contain any cycles.

3.7.1 AT-GDH Initial Key Agreement

The AT-GDH initial key agreement protocol as given in [89], executes in two phases as shown in Fig. 3.11 to 3.14:

Phase-1.

Round-1. For all nodes $x = y.i$ with $c_x = 0$

(a) x generates a random secret $k_x \in \mathbb{Z}_q$.

(b) $x \longrightarrow y : \alpha^{k_x}$.

Round-i($i \in \{2, h\}$). For all nodes x with $c_x \neq 0$

(a) x generates a random secret $e_x \in \mathbb{Z}_q$.

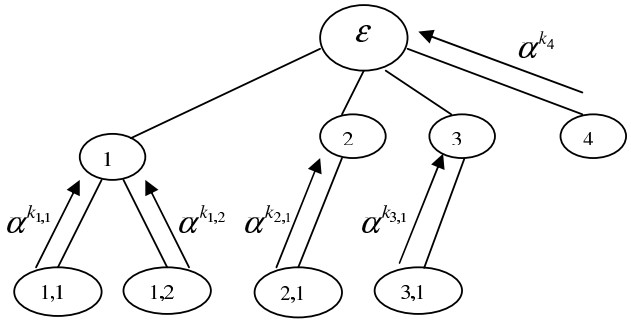

Figure 3.11: Phase 1, Round -1: For all nodes with $C_x = 0$ (i.e. has no children)

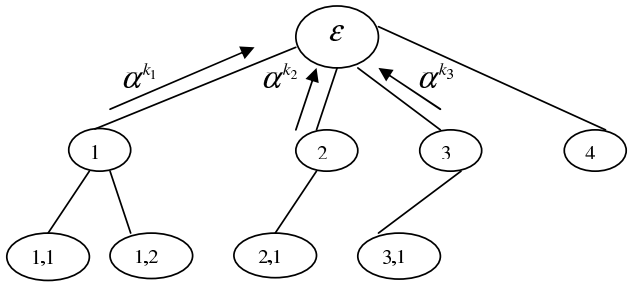

Figure 3.12: Phase 1, Round-i: Only nodes with children

(b) x waits to receive $\alpha^{k_{x.j}}$ for all $j = 1, \ldots, c_x$.

(c) x computes $k_x = \varphi(K(x, c_x))$ from:

$$K(x, 0) = e_x$$

$$K(x, j) = \alpha^{k_{x.j}\varphi(K(x,j-1))} \ (j = 1, \ldots, c_x)$$

(d) $x \longrightarrow y : \alpha^{k_x}$.

Phase-2.

Round-$(h + l)$ ($l = 1, \ldots, h$). For all nodes $x.i$ on *level* l, $x \longrightarrow x.i$: $m_{x.i}$ where

$$m_{x.i} =< m_x, \alpha^{\varphi(k(x,i-1))}, \alpha^{k_{x.(i+1)}}, \alpha^{k_{x.(i+2)}}, \ldots, \alpha^{k_{x.c_x}} >,$$

with m_ϵ being empty.

The resulting group key is $K(\epsilon, c_\epsilon) = k_\epsilon$.

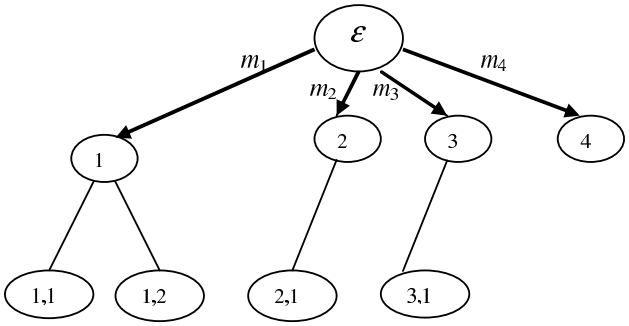

Figure 3.13: Phase 2, Round -1: For all nodes with $C_x = 0$ (i.e. has no children)

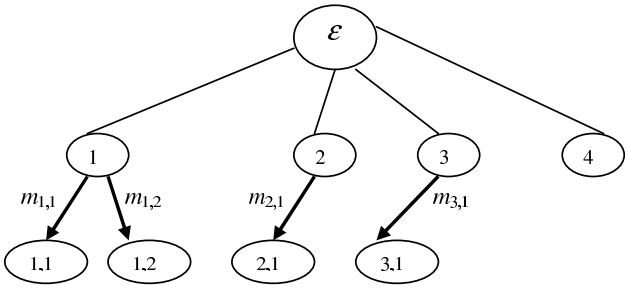

Figure 3.14: Phase 2, Round-i: Only nodes with children

3.7.2 Discussion of the AT-GDH Initial Key Agreement Protocol

The AT-GDH protocol alleviates the dependence on network topology at group *genesis*. While this approach is feasible for IKA operations, it is claimed here that AT-GDH cannot be extended to incorporate efficient AKA operations. Due to node mobility the network topology of ad hoc networks is dynamic and therefore quickly amortizes the advantage of constructing the spanning tree according to the arbitrary network topology. The constructed spanning tree will only be relevant at the time of IKA and will not reflect the network topology during the AKA operations at a later point in time. AT-GDH is thus not suitable for *mobile* ad hoc networks, but rather for ad hoc networks that become static after network deployment, for example in some applications of *sensor networks* [90].

The second notion that supports AT-GDH's inefficiencies when extended to incorporate AKA operations, is the fact that the key structure of AT-GDH protocol in its first phase is equivalent to the STR protocol proposed by [91]:

$$K_n = \alpha^{k_n \alpha^{k_{n-1} \alpha \ldots k_3 \alpha^{k_1 k_2}}} \tag{3.6}$$

The efficiency of AT-GDH's AKA operations will thus be comparable to STR. AT-GDH will be better suited for adding new members in a static network topology that is not known prior to network formation. Membership exclusion will, on the other hand, be problematic especially if the leaving member's secret exponent is used in the innermost key computation [1].

AT-GDH does not distribute the computational and communication overhead equally, although some attempt is made to perform most of the computations during the *upflow* stage in Phase 1 of the IKA protocol. AT-GDH fails to preserve the symmetric peer relationship between members, which is essential in ad hoc networks. Successful execution of the protocol also depends heavily on the root node and its children, whose presence cannot be assumed in ad hoc networks. Considering the nature and characteristics of ad hoc networks in addition to the frequent AKA operations of DPGs, any dependence on the availability of any group member will not suffice.

3.8 Contributory Key Agreement: Star Topology

Augot et al. [92] present a contributory key agreements protocol where a group leader is responsible for collecting the key contributions from members. The Initial Key Agreement (IKA) protocol resolves some of the issues of the modified Centralized Key Distribution (CKD) protocol, for example, the group leader do not have to establish a secure channel with each of the group members and cannot control the group key. See Section 3.11 for an overview and discussion on the modified CKD protocol.

The IKA group key agreement (GKA) protocol takes 3 rounds to execute:

Round 1. The chosen group leader sends an initial request (INIT) along with his identity and a random nonce to all group members.

Round 2. Each group member responds to the leader with his identity, random nonce and a blinded secret. The blinded secret is calculated by raising the agreed generator (g) to the member's random secret.

Round 3. The group leader collects all the blinded secrets and raises each to its own secret. These calculated values are then broadcast along with the original contributions (blinded secrets) to all group members. The group members finally use these values to calculate the group key.

Before continuing it would be beneficial to take note of the following additional notation:

G —subgroup with prime order q and generator g.

U_i —protocol participant i among the n group participants.

U_l —current group leader.

r_i —the random secret of U_i chosen from $[1, q-1]$.

g^{r_i} —blinder secret of U_i.

M —set of indices of participant (group M) in the current session.

J —set of indices of joining participant.

D —set of indices of leaving participant.

$x \longleftarrow y$ —x is assigned y.

$x \longleftarrow^r y$ —x is assigned a random draw from the uniform distribution S.

$U_i \longrightarrow U_j : \{msg\}$ —message msg sent unicast from participant i to j.

$U_i \longrightarrow^B M : \{msg\}$ —message msg broadcast from participant i to M.

msg_i^j —message j sent from participant i.

σ_i^j —signature on msg_i^j.

3.8.1 GKA Initial Key Agreement Protocol

The GKA IKA protocol is a natural extension of the 2-party Diffie-Hellman (DH) protocol [71]; the group leader is effectively executing a modified DH protocol with each of the other protocol participants. The group key is calculated in such as way that the contributions of all the participants are included.

The IKA protocol in [92], as illustrated in Fig. 3.15, is as follows:

Round 1. Initial Request

$$l \longleftarrow^r M, N_l \longleftarrow^r \{0,1\}^k$$

$$U_l \longrightarrow^B M : \{msg_l^1 = \{INIT, U_l, N_l\}, \sigma_l^1\}$$

Round 2. Contribution Collection

$$\forall i \in M \setminus \{l\}, if(V_{PK_l}\{msg_l^1, \sigma_l^1\} == 1), r_i \longleftarrow^r [1, q-1], N_i \longleftarrow^r \{0,1\}^k,$$

$$U_i \longrightarrow U_l : \{msg_i^1 = \{IREPLY, U_l, N_l, U_i, N_i, g^{r_i}\}, \sigma_l^1\}$$

Round 3. Shares Distribution

$$r_l \longleftarrow^r [1, q-1], \forall i \in M \setminus \{l\}, if(V_{PK_l}\{msg_l^1, \sigma_l^1\} == 1),$$

$$U_i \longleftarrow^B M : \{msg_l^2 = \{IGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{\forall i \in M \setminus \{l\}}\}, \sigma_l^2\}$$

Key Computation.

$$if(V_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$$

and g_{r_i} is as contributed then

Group Key $= g^{r_l} * \prod_{i \in M \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in M \setminus \{l\}} r_i)}$

3.8.2 GKA Auxiliary Key Agreement Protocol: Join/Merge

The GKA join/merge protocol may execute under a different group leader due to the dynamic nature of ad hoc networks. If a new group leader is elected then the old group leader has to send all the participants' blinded secrets to the new group leader. The members joining the group broadcast their blinded secrets to the group along with a JOIN message. The new group leader raises all blinded secrets to its own, newly generated secret and broadcast the results to the group, which is then used to calculate the group key.

Round 1. Join Request

$$\forall i \in J, r_i \longleftarrow^r [1, q-1], N_i \longleftarrow^r \{0,1\}^k$$

$$U_l \longrightarrow^B M : \{msg_i^1 = \{JOIN, U_i, N_i, g^{r_i}\}, \sigma_l^1\}$$

Round 2. Contribution Transfer

$$\forall i \in J, if(V_{PK_i}\{msg_i^1, \sigma_i^1\} == 1), r_i \longleftarrow^r [1, q-1], l' \longleftarrow^r M \cup J$$

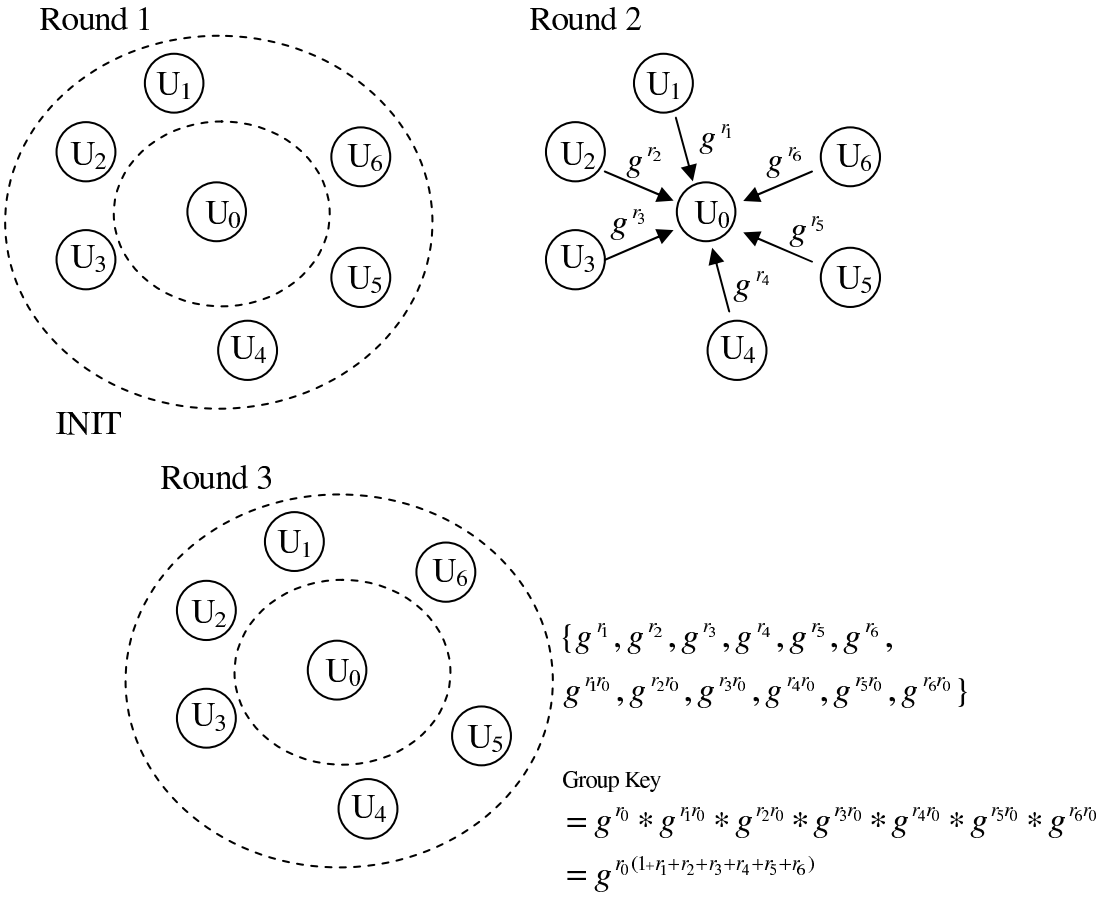

Figure 3.15: GKA IKA Key Agreement Protocol with $n = 7$ and U_0 as group leader

$$U_l \longrightarrow^B U_i : \{msg_l^1 = \{JREPLY, \{U_i, N_i, g^{r_i}\}_{\forall i \in M \cup J}\}, \sigma_l^1\}$$

Round 3. Shares Distribution

$$if(V_{PK_l}\{msg_l^1, \sigma_l^1\} == 1), l \longleftarrow l', r_l \longleftarrow^r [1, q-1], M \longleftarrow M \cup J$$

$$U_l \longleftarrow^B M : \{msg_l^2 = \{JGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{\forall i \in M \setminus \{l\}}\}, \sigma_l^2\}$$

Key Computation.

$$if(V_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$$

and g_{r_i} is as contributed then

Group Key $= g^{r_l} * \prod_{i \in M \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in M \setminus \{l\}} r_i)}$

The Delete/Partition Auxiliary Key Agreement (AKA) protocol is essentially an IKA round 3 broadcast message with the leaving members' contributions omitted. It is left to the reader to see [92] for details.

3.8.3 Discussion of the GKA Protocol Suite

The GKA IKA protocol uses broadcast messages for one-to-many communication from the group leader to group members. As the members are not dependent on the location of the other group members (excluding the leader) the protocol takes on a star topology.

The IKA protocol is contributory in the sense that the group leader and members cannot predict the group key and needs the contribution of all members, raised to the secret of the group leader, to calculate the key.

The GKA IKA protocol defend against passive adversaries by extending the 2-party DH protocol and ensures the integrity of messages by signing messages using a public key infrastructure.

There is however some problems with the GKA IKA protocol that makes it unfeasible for ad hoc networks. Firstly the protocol requires the group leader to collect the contributions from all group members, which result in the group leader having to receive $n-1$ messages within a narrow timeframe. It is clear that the route discovery from all group members to the group leader will result in a significant amount of traffic (broadcast storm) on the network as some route discovery

attempt will fail due to the limited capacity of the shared wireless medium.

In [92] it is claimed that the protocol does not rely on a centralized authority. Although it is true that the group leader does not control the group key, the leader is in control of the group membership. For example, the corrupt leader can exclude a legitimate member from the group by not raising the blinder secret of the member to the secret of the leader. This allows an attacker to focus on the group leader election protocol in order to compromise the GKA protocol.

Mobile ad hoc networks are also dynamic in nature [16], that is, nodes cannot be assumed to be connected to the network at all times. The GKA AKA protocols rely on the old group leader to share the blinded secrets of the group with the new group leader during the AKA protocols. It is not clear how this will work, since the group leader will normally only be replaced if not available.

3.9 Contributory Key Agreement: Topology Independent Schemes

[68] propose a well-known group key agreement protocol family. In literature, the *topology independent* protocol that is based on a broadcast system, is referred to as the BD protocol. The BD protocol supports a constant number of rounds and inexpensive computations [73]. While the BD protocol still requires no less than $n + 2$ exponentiations per M_i, all of these but three are $\in [1, n - 1]$. This leads to significant computational savings. The BD protocol takes only two rounds to complete, but because each round requires n broadcast messages, this translates into high communication overhead. Each member (M_i) generates a random number as secret exponent and calculates a public value z_i that is broadcast to all other group members. The members compute and broadcast a key value X_i in the second round. X_i is the exponent of M_i's personal secret value N_i to the base (z_{i+1} divided by z_{i-1}). A group member who has received all n partial key values X_i can compute the conference key K_n using a *cyclic* function.

3.9.1 BD Initial Key Agreement

The BD IKA protocol executes in two *synchronous* broadcast rounds:

Round-1. Each member M_i generates a random secret N_i, computes and broadcasts $z_i = \alpha^{N_i}$.

Round-2. Each member M_i does the following.

(a) Computes and broadcasts $X_i = \{\frac{z_{i+1}}{z_{i-1}}\}^{N_i}$

(b) Computes (modulo n) the conference key:

$$K_n = z_{i-1}^{n \cdot N_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \ldots X_{i+2} \, mod \, p \tag{3.7}$$

3.9.2 *BD* Auxiliary Key Agreement: Membership Addition

[68] does not specify any protocols to support AKA operations. Investigations have shown that such AKA operations based on the *BD* IKA protocol would all be very similar. To illustrate the basic concept of these AKA operations, the *BD* membership addition protocol is defined below.

For all *BD* AKA operations at least one of the existing group members M_k has to renew its share in the conference key K by generating a new exponent N_i' whenever a member is added. This is required in order to ensure *key freshness* [1]. Assume that the new member is added as M_{n+1}. Choosing M_k to neighbor the new member would be the most efficient choice, i.e. $M_k = M_n$ or $M_k = M_1$, since it would then affect only M_{n-1} or M_2 as members not neighboring the new member during the rekeying process.

The *BD* membership addition protocol requires two synchronous broadcast rounds:

Round-1. Current member M_n and joining member M_{n+1} each compute and distribute individual contributions $z_n' = \alpha^{N_n'}$ and $z_{n+1} = \alpha^{N_{n+1}}$.

Round-2.

(a) M_1, M_{n+1}, M_n, M_{n-1} generate and broadcast X_1', X_{n+1}', X_n', X_{n-1}' where $X_p' = \{\frac{z_{p+1}}{z_{p-1}}\}^{N_p}$, $p \in \{1, (n+1), n, (n-1)\}$

(b) Each member M_i computes (modulo n) the conference key:

$$K_n = z_{i-1}^{n \cdot N_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \ldots X_{i+2} \, mod \, p \tag{3.8}$$

where X_1, X_{n+1}, X_n, X_{n-1} is equivalent to X_1', X_{n+1}', X_n', X_{n-1}' respectively.

3.9.3 Discussion of the *BD* Protocol Suite

In [68], an efficient conference key distribution system is proposed and proved to be secure (in the pre-proceeding version [93]) provided that the *computational* Diffie-Hellman problem is intractable.

Burmester and Desmedt make important assumptions to support the feasibility of the BD protocol:

- Each member P_i has the ability to broadcast to the rest of the group.
- The system has the ability to support n simultaneous broadcast messages.
- Each member P_i has the ability to receive $n-1$ messages in a single round.

The latter two assumptions are impractical since there are no means in existing networking protocols for ad hoc networks to accommodate n simultaneous broadcasts, nor can devices receive n simultaneous messages without advanced hardware. This forces *serialized* broadcasting, which effectively eliminates the BD protocol's low round advantage. The Burmester and Desmedt protocol with serial broadcasting is referred to in literature as the BD^* protocol. Rounds 1 and 2 in the BD protocol are thus replaced in the BD^* protocol by n rounds respectively where each member M_i broadcasts its calculated z_i in round i and X_i in round $i+n$. The extra rounds in BD^* are thus due to nodes waiting for a chance to broadcast their values (z_i, X_i).

Steiner *et al* [1] point out that an extension of the BD protocol to incorporate AKA operations, may not be feasible for DPGs. Although AKA operations can be performed in only two rounds, four messages have to be received from four different sources. In comparison with the CLIQUES AKA operations this translates into high overhead [1]. [1] also claims that closer inspection reveals that all group members have to refresh their share of the group key to prevent leaking too much information or prevent members from serving as exponentiation oracles. The BD AKA operations therefore have similar computational and communication cost as the BD IKA operations making it impractical for DPGs. Also note that [68] does not specify any protocols to support AKA operations. The BD protocol is vulnerable to attacks from *active* adversaries since it provides no means of authenticating the protocol participants.

Katz and Yung [88] presents a compiler that transform the unauthenticated BD protocol into a provably secure authenticated group key agreement protocol that will withstand active adversaries.

Despite its disadvantages the BD protocol's *topology independence* makes it a very attractive option for realizing group key agreement in ad hoc networks. The topology independence aids in achieving *robustness* within a dynamic, fully distributed network setting. The BD protocol also preserves the symmetric relationship between protocol participants by distributing the burden of key management equally between all group members. With its low exponentiation cost it suits ad hoc networks' limited computational resources.

3.10 Key Pre-Distribution Schemes

In [69], a *Distributed Key Pre-distribution Scheme* (DKPS) is proposed, which eliminates the dependence of the key agreement protocol on a TTP. The proposal stems from recent research [94] [95] on *sensor networks* [90] that suggests that *key pre-distribution schemes* (KPS) are the only practical solution in scenarios where the network topology is *a priori* unknown.

DKPS is a collection of distributed cryptographic protocols that enable a number of nodes sharing no prior secret to jointly realize the key pre-distribution function. Each node individually picks a set of keys from a large publicly-known key space such that at completion, the key patterns of all the nodes satisfy the following exclusion property with high probability: any subset of nodes can find from their key patterns (in a secure manner) at least one common key that is unobtainable by a limited number of colluding nodes not belonging to the subset [69]. If a node finds that its key pattern cannot satisfy the exclusion property, after confirmation with other nodes, it will re-select another key subset. The key selection is memoryless, therefore it is assumed that the nodes will probably derive a proper key pattern on the retry. The probability of key patterns satisfying the exclusion property can be increased by feasible parameter selection. DKPS therefore has its roots in a combination of *probabilistic method* and *privacy homomorphism* [96].

The proposed DKPS scheme contains three phases:

Phase-1. Distributed key selection (DKS).

Phase-2. Secure shared-key discovery (SSD).

Phase-3. Key exclusion property testing (KEPT).

3.10.1 Distributed Key Selection (DKS)

In the DKS phase each node generates or agrees on a publicly known universal key set P and randomly picks keys to form a key-ring ($P_i \subseteq P$). This selection process satisfies an exclusion property defined as a special case of the probabilistic *cover-free family* (CFF) [69]. Note that [69] does not specifies a procedure for generating P, but specify P to be simply $\mathbb{Z}_N$.

The generalized definition of the CFF, as given in [69], is as follows:

Let P be an N-set of points $\{p_1, p_2, \ldots, p_N\}$ and β be a set of subsets X (also called blocks) of P. This can also be written as: $X \subseteq P, \forall X \in \beta$. Let N and T denote $\mid P \mid$ and $\mid \beta \mid$ respectively, then

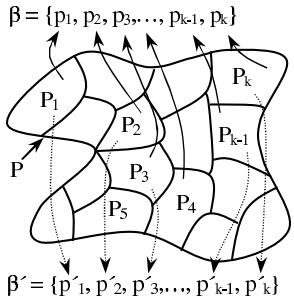

Figure 3.16: DKS based on probabilistic CFF construction

the set system (P, β) is called a $(w, r; d)$ — $CFF(N, T)$ (cover-free family) if, for any w subsets $X_1, \ldots, X_w \in \beta$ and any other r blocks $Y_1, \ldots, Y_r \in \beta$, satisfies

$$| (\bigcap_{i=1}^{w} X_i) \setminus (\bigcup_{j=1}^{r} Y_j | \geq d,$$

with d as a positive integer.

CFF is a widely adopted benchmark for formulating the security property of KPS [69]. The conventional construction methods of CFFs found in coding theory and design theory are centralized and therefore not suitable for ad hoc networks [69]. In [69], the reliance on a TTP is eliminated by constructing the CFF in a distributed manner using a probabilistic method. By constructing $(w, r; 1)$ — CFF over a publicly known universal key set P, each node or user picks a subset $X_i \subset P$ as key-ring. It follows that based on the properties of the constructed $(w, r; d)$ — CFF, any subset of nodes with size up to w can find at least one common key from their key patterns and any collusion of nodes outside this subset with size smaller than r cannot derive the common key. In Fig. 3.16 [69], it is shown how two users in a probabilistic construction can individually pick keys to form their key rings, while satisfying the exclusion property of $(w, r; d)$ — CFF.

If the capacity of each node's key-ring is denoted by k_B the DKS construction of $(w, r; d)$ — CFF is as follows:

Step-1. The nodes select $k \leq k_B$ such that d divides k.

Step-2. The universal key set P is formed with size $N = k \cdot u \cdot r$, where $u = \frac{k}{d}$. Each node holds its own instance of P.

Step-3. Key set P is divided into k partitions $P_1, P_2, \ldots, P_k$, each of size $u \cdot r$.

Step-4. The nodes individually pick keys for their rings to form $\beta = \{p_1, p_2, \ldots, p_k\}$, with each p_i randomly selected from the partition $P_i, 1 \leq i \leq k$.

3.10.2 Secure Shared-key Discovery (SSD)

In the key discovery phase all nodes pairwise establish which keys in their key-rings they have in common, without revealing information about the keys not in common.

The SSD protocol is based on *private homomorphic encryption* (PH encryption) [97] [98] [99]. In [69], the author introduces a *Modified Rivest's Scheme* (MRS) based on the original algorithm by [96] to eliminate problems identified in other PH encryption schemes.

In conventional KPS the TTP gives each node a corresponding key identifier for each key in the node's key-ring. The TTP also knows the mapping between all keys and identifiers. The TTP reveals to the nodes the mapping between the keys and identifiers relevant only to the keys on the node's key-ring. In this case the nodes can easily initiate SSD by broadcasting their key identifiers to other protocol participants. Consequently only those nodes that have the same keys can gain knowledge of the keys they have in common.

For a system without a TTP, SSD is not so trivial. The DKPS proposed in [69] implements an SSD protocol using MRS. The SSD or *secure set intersection problem* is defined by [69] as follows:

- Two parties, Alice and Bob, have two key sets $A = \{a_1, a_2, \ldots, a_l\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ respectively. How can two parties obtain $A \cup B$ without allowing other parties to learn the elements outside the intersection? The exchange of $A \cup B$ must also be secure from *passive* adversaries, eavesdropping on the SSD procedure.

The SSD-MRS protocol [69] attempts to solve the SSD problem in four steps:

Step-1. Alice forms a polynomial, $f_A(x) = x^l + A_{l-1}x^{l-1} + \ldots + A_1x + A_0$ and encrypts the coefficient using $E_{K_A}(\cdot)$ where K_A is her keys in subset A. These are transferred to Bob.

Step-2. Bob sorts his key set B in descending order and chooses a random secret r_B. The homomorphic properties of $E_{K_A}(\cdot)$ allows Bob to compute $z_i = E_{K_A}(r_B f_A(B))$ for $1 \leq i \leq m$

Step-3. Alice receives all z_i's and decrypts them using K_A. Alice thus obtains $r_B f_A(B)\ \forall\ 1 \leq i \leq m$. Because B are blinded by r_B, Alice gains no knowledge of Bob's keys. What Alice knows is $B \in A$ that satisfy $r_B f_A(B) = 0$.

Step-4. Alice returns a m-bit bitmap with 1 at the bits where $r_B f_A(B) = 0$. All common keys in $A \cup B$ can be used to compute a session key only known by Alice and Bob.

Note that the SSD-MRS must be executed in parallel from both parties' perspective. The above description applies to Bob checking with Alice which keys they have in common.

3.10.3 Key Exclusion Property Testing (KEPT)

The third and final phase of the DKPS scheme ensures that the exclusion property holds. If not, the protocol has to be re-executed from phase 1 with DKS followed by SSD-MRS in phase 2.

Each node can represent the pattern of how its keys are shared with other nodes in a binary structure called an *incidence matrix*. Using an incidence matrix each node can verify whether all keys satisfy the exclusion property of $(w, r; d)$ — CFF. The definition of an incidence matrix is given in [69] as follows: if a node has k keys in its key-ring and there are m other nodes then its incidence matrix is a $m \times k$ binary matrix $A = [a_{ij}]$ with:

$$a_{ij} = \begin{cases} 1 & \text{, if the } j^{th} \text{ key is shared with the } i^{th} \text{ node} \\ 0 & \text{, otherwise} \end{cases}$$

If the exclusion property holds then a vector formed by the bitwise-AND operation of any w rows of A will have at least d bit positions of '1' different from the vector formed by taking the bitwise-OR of any other r rows.

3.10.4 Computing the Group Key

After completing the DKPS protocols, each node can find a block of keys in common with any group of other nodes. The subset of keys are used by all group members to compute the group key known to all by taking a hash function $h(\cdot)$ of the concatenation of all the common key's k_i.

$$K_{group} = h(k_1 \parallel \cdots \parallel k_i \parallel \cdots \parallel k_x) \tag{3.9}$$

3.10.5 Discussion of the Distributed Key Pre-Distribution Scheme

In [69], asymmetric/public key cryptosystems are deemed unsuitable for ad hoc networks due to their inefficiency. The author of [69] refers to [76], an investigation into key management schemes for *sensor networks*, to substantiate this statement. While this statement is accurate for sensor

networks, this is not the case for ad hoc networks. Although having similar constraints to ad hoc networks, sensor networks have much tighter constraints on the nodes' energy, computational and memory resources. The nodes in sensor networks, as a result of their intended application, are very small with dimensions in the cubic millimeter range [100]. These 'stringent' constraints on energy, computational and memory resources makes asymmetric schemes impractical for use in sensor networks [94] [95]. Although nodes in ad hoc networks have limited resources, these resources are sufficient to support asymmetric cryptosystems [3] [4] [35]. This is in particular true with recent advances in elliptic curve cryptography [65] [101].

From the discussion on the distributed key selection (DKS) protocol (Section 3.10.1), it should be clear that any node that has the universal publicly known key set P, can randomly pick keys to form an individual key-ring. By the properties of the cover-free family (CFF), $(w, r; 1)$ — CFF, the adversary executing the DKS protocol shares at least one common key with all of the nodes in the network. The SSD-MRS protocol given in Section 3.10.2 allows users to discover what keys they have in common with other nodes, while satisfying the exclusion property. The SSD-MRS protocol offers no mechanism to provide protocol participants with *implicit* key authentication, i.e., a member M_i of a group M has no assurance that an adversary $M_q \notin M$ cannot impersonate a legitimate M_i and learn the common group keys and therefore compute K_{group}. DKPS is intended for nodes forming a group with no prior contact or pre-distributed shared secrets, except a publicly known pool of keys P [69]. With no prior secret share between the group members as stated in [69], it is not clear how DKPS will achieve implicit key authentication. In a truly *ad hoc* setting nodes may initially not be concerned with the identities of the other nodes since these nodes have no relationships prior to group formation. Nevertheless, a mechanism is required to uniquely bind a node to its identity in order to hold a node accountable for its actions. It should be clear that as these nodes build relationships after group *genesis* an authentication mechanism will become essential.

An *authenticated group key agreement protocol* (AGKAP) allows group members to establish a shared secret key in the presence of an *active* adversary [11] [75]. Providing *key authentication* using DKPS as a stand alone symmetric group key management scheme, without a TTP, is not possible. It is well known that asymmetric key cryptosystems require computational and energy resources that are orders of magnitude more than symmetric key cryptosystems. Despite the inefficiency, asymmetric cryptosystems have properties making them superior for distributing keys, providing authentication, data integrity and non-repudiation [11] [3]. The latter three of these functionalities provided by asymmetric cryptosystems play an important role in group key management protocols. The most feasible security mechanisms will therefore incorporate a hybrid cryptoscheme drawing

from the advantages of both symmetric and asymmetric based schemes.

The security of DKPS rests on the probability of the system satisfying the criteria of a cover-free family (CFF). As explained in [69], most of the nodes would successfully establish a good key set satisfying the exclusion property. Although there is a high probability of success, it is in fact only known if the exclusion property holds after execution of the key exclusion property testing (KEPT) protocol. If the exclusion property is violated then the two protocol participants need to re-execute the DKPS protocols until the KEPT protocol returns a positive result.

Suppose a node M_1 in the initialization phase of the network performs DKS and SSD with M_2 and SSD with each of the other $n-2$ nodes in the network. After each execution of the DKS and SSD-MRS protocols, KEPT is used to test if the exclusion property holds. Assume that KEPT returns a positive result for the first $i-2$ nodes, but returns a negative result for the common keys shared between M_1 and M_i. What this implies is that the whole process must start over as M_1 has to re-execute the DKS protocol and consequently SSD-MRS with all the other nodes for which KEPT has already returned a positive result. It is thus not clear if the DKPS scheme will converge and yield good key sets that satisfy the exclusion property for all node combinations in the network. If convergence is possible then DKPS requires all n group members to perform DKS once, twice or even more times, with SSD followed by the execution of KEPT with the remaining $n-1$ members. Without optimization, this will result in a best case latency of $n(n-1)$ DKPS executions before K_{group} can be calculated. Robust implementation and practicality of DKPS may therefore be problematic.

It is noted here that DKPS fails to provide members of dynamic peer groups with *key freshness*, *perfect forward secrecy*, *key independence* and resistance to *known key attacks* as defined in [16]. It can thus be concluded that DKPS is not suitable for DPGs found in ad hoc networks.

3.11 Decentralized Key Distribution

The *Centralized Key Distribution* (CKD) protocol suite as given in [74] is not suitable for DPGs. In order to satisfy the properties of general *decentralized* key distribution protocols, as defined in Section 3.1, only one modification was made to the CKD scheme. As illustrated in Fig. 3.1, *centralized* key management is a subset of *key transport* that is normally used by large *non-collaborative* groups. The primary difference between centralized and decentralized key transport protocols is that centralized schemes have a designated TTP responsible for all initial key transport (IKT)

operations and auxiliary key transport (AKT) operations, while these operations are performed by any randomly chosen group member in the decentralized schemes. In the case of the CKD scheme [74], the TTP (group controller) is always the oldest group member. By allowing the role of group controller to be randomly assigned to any group member, the CKD scheme as presented in [74] can be grouped with decentralized key agreement schemes as opposed to centralized schemes. The *modified* CKD scheme will thus randomly select any group member to take on the responsibility of group controller at the beginning of each AKT operation.

Regardless of the group event (IKT or AKT operation) the protocol executes in two phases:

Phase-1. Each group member and the randomly chosen group controller agree on a symmetric key using the authenticated II-party *DH* protocol [102]. On each AKT operation the new group controller has to repeat this operation with every member.

Phase-2. The group controller independently generates and distributes the group key. For this reason CKD can be seen as a *key transport* protocol.

3.11.1 CKD Auxiliary Key Transport: Mass Join

The CKD *mass join* protocol adds k members to a group of size n. Assume member M_1 to be the randomly chosen group controller.

The CKD mass join protocol executes in four steps:

Step-1. M_1 selects a random secret $r_1 \in \mathbb{Z}_q$.

$$M_1 \longrightarrow \{M_{n+j} | j \in [1,k]\} : \alpha^{r_1} \ mod \ p$$

Step-2. For each $j \in [1,k]$, M_{n+j} selects a random secret $r_{n+j} \in \mathbb{Z}_q$.

$$M_1 \longleftarrow M_{n+j} : \alpha^{r_{n+j}} \ mod \ p$$

Step-3. M_1 selects a random group secret r_{gs}.

$$M_1 \longrightarrow M_i : r_{gs}^{\alpha^{r_1 r_i}} \ mod \ p \ \forall \ i \in [2, n+k]$$

Step-4. All group members $n+k$ compute K' from the broadcast message.

3.11.2 CKD Auxiliary Key Transport: Mass Leave

The CKD *mass leave* protocol excludes l members from a group of size n. Assume members M_1 to be the randomly chosen group controller.

The CKD mass leave protocol executes in two steps:

Step-1. M_1 selects a new group random secret r'_{gs}

$$M_1 \longrightarrow M_i : r'^{\alpha^{r_1 r_i}}_{gs} \ mod \ p, M_i \notin l$$

Step-2. All group members $n - l$ compute K' from the broadcast message.

3.11.3 Discussion of the CKD Protocol Suite

Some features of decentralized key distribution protocols make them unsuitable for ad hoc networks:

1. The central entity (group controller) presents a single point of vulnerability although it is randomly chosen among the group members.
2. On each AKT operation, the new randomly chosen group controller is required to perform an authenticated II-party DH key exchange with each group member, which results in significant additional communication and computational overhead.
3. Ad hoc networks are highly dynamic in nature and no group member can be assumed to be present all the time. The central group controller may be unavailable due to any number of factors such as moving out of transmission range, error-prone wireless connectivity or depleted resources. In the event of the controller not being available a new controller must take over the responsibility, which results in additional high overhead.
4. Centralized key distribution cannot provide *perfect forward secrecy*. [1].
5. Group members do not get assurance that the group key is random and satisfy *key freshness* [1].

As pointed out in [1], a central point is however needed for the administration of group membership operations. The functionality of group controller should be performable by any group member. The group controller serves only to synchronize the IKA and AKA operations to prevent confusion. It

should thus be clear that the existence and assignment of the group controller should be orthogonal to the key agreement protocols and depend solely on group policy.

3.12 Performance Analysis

The computational and communication overhead of *contributory* key agreement protocols are important issues to consider when determining the protocols' suitability for ad hoc networks. This is due to ad hoc networks' limited computational and communication resources. Table 3.1 to Table 3.5 compare the *worst case* communication cost and computational cost of the most promising existing schemes (CLIQUES [1], $TGDH$ [9] and BD^* [68]) that incorporate IKA and AKA operations.

3.12.1 Efficiency of IKA operations

Table 3.1 shows that in terms of communication cost, CLIQUES IKA.1 and IKA.2 protocols outperform the BD^* protocol. Although the BD protocol executes in only two rounds, a *synchronous* broadcast system is impractical (Section 3.9.3). The BD^* protocol's $2n$ broadcast messages would result in more network congestion than CLIQUES IKA.1 and IKA.2, which predominantly use *unicast messages*. In the BD^* protocol's defence it should be realized that in order to place an equivalent amount of overhead on each group member in a fully distributed, contributory system, the scheme is bound to be based on a broadcast system. In such a scenario broadcast is not only the most efficient means of communication, but has been proven to be the most reliable in ad hoc networks [35]. The reader is also referred to [67], which defines the lower bound on message exchange of a contributory, broadcast based key agreement scheme to be n broadcast messages. The BD^* protocol thus requires its communication overhead to be halved in order to be optimal.

The BD and BD^* protocols' key construction procedure allows for significant savings in computational cost. As illustrated in Table 3.2, of the $n+2$ exponentiations performed by each member, $n-1$ of these are 'low cost' (although the additional time complexity added by these smaller exponentiations may not be insignificant for larger group sizes). From the computational efficiency analysis results given in Table 3.2 through to Table 3.5 it is clear that CLIQUES fails to distribute the burden of key management fairly between all group members. [1] improves on the computational overhead distribution of CLIQUES IKA.1 with IKA.2, reducing the exponentiations per protocol participant to only four. There may however be a few inherited problems with this

enhancement if CLIQUES IKA.2 protocol were to be used in ad hoc networks:

- The highest indexed member (M_n) is still required to perform n exponentiations. The improvement thus will not aid in mitigating *selfishness* attacks [77].
- The improvement from IKA.1 to IKA.2, in terms of computational overhead distribution, comes at the cost of significant communication overhead.
- The highest indexed member's role in the initial key agreement protocol becomes centralized, giving adversaries a single point of attack.

The computational cost introduced by message authentication (which implicitly implies member authentication) is given in Table 3.4 and Table 3.5 in terms of the number of signatures generated and signatures verified by each group member. The number of signatures generated by members during initial key agreement is very similar for all protocols. The major difference relates to the number of verifications per member. Here the CLIQUES IKA.1 and IKA.2 protocols [1] prove to be superior to the BD^* IKA protocol [68]. One should however consider that signature schemes such as RSA, provides significantly faster signature verification than signature generation procedures [11].

3.12.2 Efficiency of AKA operations

In DPGs, efficient AKA operations are more important than IKA operations, since the latter is performed only once [1]. It is therefore important for key management schemes, designed for ad hoc networks, to minimize the communication and computational overhead associated with the AKA operations (Fig. 3.2).

As mentioned in Section 3.9.3, [1] claims that closer inspection reveals that all group members have to refresh their share of the group key on each group event in order to prevent leaking too much information or prevent serving as exponentiation oracles. Since this is not formally proved in literature, the efficiency analysis of the BD^* protocols AKA operations is given for the optimal case where only one member is required to renew its share in the group key.

To make a feasible comparison between the AKA operations of the existing schemes [1] [9] [68], communication and computational overhead have to be considered simultaneously. An overall assessment shows that *tree-based* protocols ($TGDH$) are superior to the *serial topology* protocols (CLIQUES) and *topology independent* protocols (BD^*) with respect to changes in group membership. Tree-based protocols are thus more easily scalable and better suited for large groups

($n > 100$). In a dynamic peer group setting, group sizes will however be small, which limits the improved performance of tree-based schemes. In order to make an overall performance comparison between the existing schemes, the *time complexity* of the respective AKA operations might be a more feasible parameter to consider. Since the time complexity is dependent on numerous factors (such as node processing power, network environment, group size, and number of members leaving or joining) a more realistic study is required through simulation. It is however noted that since the existing schemes [1] [9] [68] are unsuitable for ad hoc networks, the conclusions drawn from such simulations with respect to efficiency comparisons may be impractical or limited. Simulations and a performance comparison on the existing group key management schemes in a *conventional* network setting have been presented by Amir *et al.* and the reader is referred to [74] for details.

3.13 Conclusions

This article presents a survey on the published group key management schemes with respect to their suitability for mobile ad hoc networks. Studies within the available literature have shown that group-oriented communication in ad hoc networks will generally occur in the form of *dynamic peer groups* (DPGs). The protocols for DPG settings were categorized in terms of their underlying key establishment mechanisms. These categories included.

- *Contributory key agreement* and *key pre-distribution* as subsets of *key agreement*.
- *Decentralized* key distribution as a subset of *key transport*.

Contributory key agreement was further uniquely subdivided based on the topology enforced upon the group members as a derivative of the construction method used to obtain a desired group key form. Each of these subsets was discussed by introducing at least one group key management scheme from within the grouping. The analysis of each group key management scheme was followed by a discussion on the scheme's suitability for ad hoc networks. The performance of the protocol suites that consider both *initial key agreement* (IKA) and *auxiliary key agreement* (AKA) operations was presented and analyzed.

The conclusion drawn from the survey is that distinct challenges arrive when adapting group key management protocols designed for conventional networks to suit ad hoc networks:

- Protocols cannot be dependent on any specific order of nodes participation or hierarchical structure (topology) due to ad hoc networks' dynamic network topology.

Table 3.1: Communication cost comparison between reviewed protocol suites

Initial Key Agreement Protocols		Rounds	Messages	Unicast	Broadcast
Cliques	IKA.1	n	n	$n-1$	n
	IKA.2	$n+1$	$2n-1$	$2n-3$	2
$TGDH$		n/a	n/a	n/a	n/a
BD		2	$2n$	0	$2n$
BD^*		$2n$	$2n$	0	$2n$
Auxiliary Key Agreement Protocols		**Rounds**	**Messages**	**Unicast**	**Broadcast**
Cliques[a]	MA	2	2	1	1
	ME	1	1	0	1
	MJ	$u+1$	$u+1$	u	1
	ML	1	1	0	1
$TGDH$[b]	MA	2	3	0	3
	ME	1	1	0	1
	GM	$\lceil log_2 k \rceil + 1$	$2k$	0	$2k$
	GP	$min\left[(log_2 v + 1), h\right]$	$min\left[2v, \lceil \frac{n}{2} \rceil\right]$	0	$min\left[2v, \lceil \frac{n}{2} \rceil\right]$
(BD^*)[a c]	MA	5	5	0	5
	ME	4	4	0	4
	MJ	$u+4$	$u+4$	0	$u+4$
	ML	$2(w+c+1)$	$2(w+c+1)$	0	$2(w+c+1)$

[a] u denotes the number of members joining the group.

[b] k denotes the number of subgroups, v the number of leaving members and h the height of the key tree.

[c] $w = \{M_i\}$, where each M_i represents an isolated leaving member. $c = \{c_j\}$ represents the set of leaving subgroups where $c_j = \{M_j, M_{j+1}, \ldots, M_{j+k}\}$, for $k \geq 1$ and $j \in \beta$, denotes a subgroup of indexed neighboring members leaving together. The total number of leaving members is thus $v = w \cup c$. The total number of rounds is given for $i - j > 1$. For the sake of simplicity the case when $i - j = 1$ will not be considered in this table.

Table 3.2: Computational cost comparison between reviewed protocol suites with respect to exponentiations per group member

Initial Key Agreement Protocols		Exponentiations per M_i
Cliques	IKA.1	$(i+1)$ for $i < n, n$ for M_n
	IKA.2	4 for $i < (n-1), \{2, n\}$ for $\{M_{n-1}, M_n\}$
$TGDH$		n/a
BD		$n+2$
BD^*		$n+2$
Auxiliary Key Agreement Protocols		**Exponentiations per M_i**
Cliques[a]	MA	$\{n+1, n+2\}$ for $\{M_c, M_{n+1}\}, 1$ for other M_i
	ME	$(n-1)$ for $M_c, 1$ for other M_i
	MJ	$(i+n)$ for $M_{i+n}, 0 \leq i < u, (u+n)$ for $M_{n+u}, 1$ for other M_i
	ML	$(n-v)$ for $M_c, 1$ for other M_i
$TGDH$[b]	MA	$2(h-1)$ for $M_s, (h-1)$ for other M_i
	ME	$2(h-1)$ for $M_s, (h-1)$ for other M_i
	GM	$2(h-1)$ for $M_s, (h-1)$ for other M_i
	GP	$2(h-1)$ for $M_s, (h-1)$ for other M_i
(BD^*)[a c]	MA	$(n+1)^e$ for $M_i,\ i \in \{(n-1), n, (n+1), 1\}, n^e$ for other M_i
	ME	$(n+1)^e$ for $M_i,\ i \in \{(j-2), (j-1), (j+1)\}, n^e$ for other M_i
	MJ	$(n+1)^e$ for $M_i,\ i \in \{(n-1), n, \ldots, (n+u), 1\}, n$ for other M_i
	ML	$(n+1)^e$ for $M_i\ i \in \sigma$;[d] n^e for other M_i

[a] u denotes the number of members joining the group.

[b] k denotes the number of subgroups, v the number of leaving members and h the height of the key tree.

[c] $w = \{M_i\}$, where each M_i represents an isolated leaving member. $c = \{c_j\}$ represents the set of leaving subgroups where $c_j = \{M_j, M_{j+1}, \ldots, M_{j+k}\}$, for $k \geq 1$ and $j \in \beta$, denotes a subgroup of indexed neighboring members leaving together. The total number of leaving members are thus $v = w \cup c$. The total number of rounds is given for $i - j > 1$. For the sake of simplicity the case when $i - j = 1$ will not be considered in this table.

[d] Members neighboring w, c and M_k, form the set σ.

[e] $n-1$ of these exponents are 'low' cost, i.e. $\in [n-1]$.

Table 3.3: Computational cost comparison between reviewed protocol suites with respect to total exponentiations

Initial Key Agreement Protocols		Total Exponentiations
Cliques	IKA.1	$\frac{n(n+3)}{2} - 1$
	IKA.2	$5n - 6$
$TGDH$		n/a
BD		$n(n+2)$
BD^*		$n(n+2)$
Auxiliary Key Agreement Protocols		**Total Exponentiations**
Cliques[a]	MA	$3n + 2$
	ME	$2n - 3$
	MJ	$\frac{u}{2}(u+1) + n(u+2) - 1$
	ML	$2(n-v) - 1$
$TGDH$[b]	MA	$3h - 3$
	ME	$3h - 3$
	GM	$3h - 3$
	GP	$3h - 3$
(BD^*)[a] [c]	MA	$n^2 + n + 4$
	ME	$n^2 - n + 3$
	MJ	$n^2 + u(n-1) + 3$
	ML	$n^2 - vn + 2(w+c) + 1$

[a] u denotes the number of members joining the group.

[b] k denotes the number of subgroups, v the number of leaving members and h the height of the key tree.

[c] $w = \{M_i\}$, where each M_i represents an isolated leaving member. $c = \{c_j\}$ represents the set of leaving subgroups where $c_j = \{M_j, M_{j+1}, \ldots, M_{j+k}\}$, for $k \geq 1$ and $j \in \beta$, denotes a subgroup of indexed neighboring members leaving together. The total number of leaving members are thus $v = w \cup c$. The total number of rounds is given for $i - j > 1$. For the sake of simplicity the case when $i - j = 1$ will not be considered in this table.

Table 3.4: Computational cost comparison between reviewed protocol suites with respect to signatures per group member

Initial Key Agreement Protocols		Signatures per M_i
Cliques	IKA.1	1
	IKA.2	1 for M_n, 2 for other M_i
$TGDH$		n/a
BD		2
BD^*		2
Auxiliary Key Agreement Protocols		**Signatures per M_i**
Cliques[a]	MA	1 for $\{M_c, M_{n+1}\}$
	ME	1 for M_c
	MJ	1 for $M_i, i \in [n, n+1, \ldots, n+u]$
	ML	1 for M_c
$TGDH$[b]	MA	2
	ME	1
	GM	$\lceil log_2 k \rceil + 1$
	GP	$min\,[(log_2 v + 1), h]$
(BD^*)[a] [c]	MA	1 for $M_i,\ i \in \{(n-1), (n+1), 1\}$, 2 for M_n
	ME	1 for $M_i,\ i \in \{(j-2), (j+1)\}$, 2 for M_{j-1}
	MJ	1 for $M_i,\ i \in \{(n-1), (n+1), \ldots, (n+u), 1\}$, 2 for M_n
	ML	1 for $M_i\ i \in \sigma$,[d] 2 for M_k

[a] u denotes the number of members joining the group.

[b] k denotes the number of subgroups, v the number of leaving members and h the height of the key tree.

[c] $w = \{M_i\}$, where each M_i represents an isolated leaving member. $c = \{c_j\}$ represents the set of leaving subgroups where $c_j = \{M_j, M_{j+1}, \ldots, M_{j+k}\}$, for $k \geq 1$ and $j \in \beta$, denotes a subgroup of indexed neighboring members leaving together. The total number of leaving members are thus $v = w \cup c$. The total number of rounds is given for $i - j > 1$. For the sake of simplicity the case when $i - j = 1$ will not be considered in this table.

[d] Members neighboring w, c and M_k, form the set σ.

Table 3.5: Computational cost comparison between reviewed protocol suites with respect to verifications per group member

Initial Key Agreement Protocols		Verifications per M_i
Cliques	IKA.1	1 for $\{M_1, M_n\}$, 2 for other M_i
	IKA.2	2 for $\{M_1, M_{n-1}\}$, $(n-1)$ for M_n, 3 for other M_i
$TGDH$		n/a
BD		$2n$
BD^*		$2n$
Auxiliary Key Agreement Protocols		**Verifications per M_i**
Cliques[a]	MA	1
	ME	1 for $M_i, i \neq c$
	MJ	2 for $M_i, i = [(n+1), (n+u-1)]$, 1 for other M_i
	ML	1 for $M_i, i \neq c$
$TGDH$[b]	MA	3
	ME	1
	GM	$\lceil log_2 k \rceil$
	GP	$min[2v, \lceil \frac{n}{2} \rceil]$
(BD^*)[a c]	MA	3 for M_n, 4 for other M_i
	ME	2 for M_{j-1}, 3 for other M_i
	MJ	$u+2$ for M_n, $u+3$ for other M_i
	ML	$2(w+c)$ other $M_i, i \in \sigma$;[d] $2(w+c)+1$ other M_i

[a] u denotes the number of members joining the group.

[b] k denotes the number of subgroups, v the number of leaving members and h the height of the key tree.

[c] $w = \{M_i\}$, where each M_i represents an isolated leaving member. $c = \{c_j\}$ represents the set of leaving subgroups where $c_j = \{M_j, M_{j+1}, \ldots, M_{j+k}\}$, for $k \geq 1$ and $j \in \beta$, denotes a subgroup of indexed neighboring members leaving together. The total number of leaving members are thus $v = w \cup c$. The total number of rounds is given for $i - j > 1$. For the sake of simplicity the case when $i - j = 1$ will not be considered in this table.

[d] Members neighboring w, c and M_k, form the set σ.

- Integration of a robust authentication mechanism is required to mitigate attacks from stronger *active* adversaries. It is essential that these mechanism introduce minimal additional computational and communication cost.
- The computational and communication overhead should be fairly or equally distributed to all members or participants. This is essential to mitigate *selfishness* attacks.
- The protocol should be deployable in a self-organized network and thus ideally eliminate any imbalance between the responsibilities of nodes. The protocol should therefore be *fully* distributed to avoid single points of vulnerability.
- If special responsibilities (such as group controller) are delegated to the group members, such roles should be randomly assigned to any group member based on group policy and therefore be *orthogonal* to the key establishment process.

Adhering to the latter three of these conditions allows the group key management scheme to preserve the strong symmetric relationships between the members of the DPGs found in ad hoc networks.

The final conclusion drawn from the discussions on the existing key agreement schemes for DPGs, is that only the Burmester and Desmedt (BD) *topology independent* contributory key agreement protocol [68], which is based on a broadcasting system, seems to satisfy most of the fundamental properties of ad hoc networks. According to [1], the BD key agreement protocol however suffers from some disadvantages, for example, high communication overhead and susceptibility to *active* adversaries, which makes it impractical.

This study confirms that key management mechanisms proposed to support the security of conventional networks are not necessarily suitable or adaptable to ad hoc networks. New techniques, designed specifically for ad hoc networks, are necessary. Key management for DPGs in ad hoc networks is still an interesting research area with room for innovation.

Chapter 4

Bootstrapping Group Communication and Security in Mobile Ad Hoc Networks

4.1 Introduction

Users of ad hoc networks may require to participate in collaborative applications supported by dynamic peer groups (DPGs) [1] [79] [80] [2] (Chapter 3). Conventional, large groups found, for example, with internet multicast applications are normally non-collaborative and hard to control on a point-to-point basis [1] [103] [79]. They therefore have a hierarchical structure maintained by an online authority and exhibit one-to-many broadcast communication patterns [1] [103] [9] [79]. The characteristics of ad hoc networks determine that DPGs, as small collaborative groups (with membership in the order of a hundred), may be more suited for group-oriented applications than large non-collaborative groups [103] [79] [2].

DPGs have to be supported by a reliable, *view-oriented* Group Communication System (GCS) [10] designed specifically for ad hoc networks, of which the ProbabilistIc Lightweight grOup communication sysTem (PILOT) [2], is a noteworthy contribution. The GCS infrastructure provides DPGs with important communication functions such as membership services, multicasting and data sharing [1] [10] [2]. In turn, the GCS itself may need a DPG instance to function such as the Storage

Set (StS) used by Pilot. The StS holds shared data in a replicated fashion that can be updated or queried by group members (or agents) using the data sharing service and underlying multicast protocol [2].

In mobile ad hoc networks, error-prone multihop wireless links and node mobility give rise to frequent route failures. These characteristics cause unpredictable and dynamic network topologies that make group communication vulnerable to various attacks and notoriously challenging to bootstrap and secure [3] [5] [28]. In this context the expression, *bootstrap security*, refers to the practical implementation of a set of techniques and procedures supporting the establishment and maintenance of keying material between authorized nodes, starting with no shared keying material between nodes prior to network formation and, without any assistance from an online authority.

4.1.1 Problem Statement

In this chapter we address the related problems of bootstrapping group communication and security for DPGs and the underlying GCS for ad hoc networks. The objective is to bootstrap the following GCS protocols: 1) unicast routing, 2) group membership services, 3) multicast routing, 4) group key agreement and 5) data sharing. Based on our investigations the process of starting an instance of the GCS that is self-initiating and self-sustaining (what we refer to as bootstrapping group communication) should be considered together with providing the GCS with the necessary keying material to secure the protocol structure listed above (what we refer to as bootstrapping the security of group communication). More precisely, the expression *bootstrap group communication* refers to the practical implementation of a set of techniques and procedures supporting the establishment and maintenance of a group *view* between authorized nodes, starting with no shared *view* between nodes prior to network formation and, without any assistance from an online authority. Bootstrapping of the GCS is mainly performed by the group membership service [10].

To discuss the problem in more detail we consider the characteristics of GCS' suitable for ad hoc networks and discuss the main design challenges and requirements for bootstrapping a GCS and the underlying security mechanisms.

4.1.1.1 Security Requirements for Group Communication

The minimum set of security services provided by a GCS are presented in [80]. In order to bootstrap the security for group communication, the security mechanisms in support of membership services,

multicast routing and data sharing schemes, should be provided with timely cryptographic keying material by the key management scheme [80].

Next, we consider each of the GCS architecture protocols listed above to highlight specific design challenges and requirements. The dependencies between the protocols of a GCS are important for security and should be considered when bootstrapping group communication systems.

4.1.1.2 Bootstrapping Security for Unicast Routing

The security of the unicast routing protocol should be bootstrapped first. We define the requirements for bootstrapping the security of a unicast routing protocol in [17] [16] and Chapter 2. The most difficult challenges are to break the *routing-security interdependence cycle* [33] and share only the necessary keying material without relying on a distributed online authority [16]. Keying material cannot be trivially shared between all network participants prior to network formation as this will make the network nonscalable and prevent spontaneous network formation [17]. Including keying material in routing packets is an easy way to break the routing-security interdependence cycle, but is inefficient considering the large number of control packets generated by on-demand routing protocols [36] [17].

4.1.1.3 Group Membership Service for Bootstrapping Group Communication Systems

The group membership service maintains the *view* or listing of the current active and connected members (or processes) for each group communication participant [10]. The safety and liveness properties of a membership service for GCS is defined in [10]. An efficient approach would require the key management protocol, in support of the unicast routing protocol, to provide the membership service with the necessary keying material to secure the construction and maintenance of views.

Applying conventional membership services to ad hoc networks will most likely be *partitionable* due to the difficulty of providing a *primary-partition* (or primary components) group membership service in asynchronous systems [104] [105] [2]. Given the characteristics of ad hoc networks, such as, node mobility and frequent node and route failures, these factors may prevent the membership service from maintaining a single, full membership view [2].

Contributory group key agreement protocols, as we discuss in [19], cannot tolerate multiple views

[79] [1]; a primary-partition group membership service is needed. Stronger reliability mechanisms and assumptions are required for the primary-partition group membership protocols to remain solvable in ad hoc networks [104] [105].

4.1.1.4 Bootstrapping Security for Multicasting

The multicast protocol delivers ordered messages to the current multicast-group members [80]. The safety and liveness properties of such a multicast service are defined in [10]. An efficient approach would require that all the necessary keying material to secure the multicast protocol is available as the membership service install a *view*.

4.1.1.5 Bootstrapping Security for Contributory Group Key Agreement

Most existing contributory group key agreement schemes[1], as we discuss in [19], enforce a virtual topology on the group structure by the construction method used to obtain a desired group key form. The virtual topology formed by the group members is determined by the order in which protocol participants perform operations on intermediate keying material to form the group key or is determined by the order in which the participants generate intermediate keying material. Based on our analysis [19], the group key agreement protocol will be the most efficient if the virtual structure of the group, maps to the underlying network topology formed by the nodes. In practice, the dynamic virtual structure of the group will not correspond to the dynamic network topology of the ad hoc network as the group membership changes and the users change position[2]. This situation clearly surpasses the design challenges due to dynamic membership and network faults envisioned in wired networks by [1] [79]; combining the dynamic network topology of ad hoc networks with the dynamic membership of DPGs on the application layer give rise to a system with little determinism.

Our objective is not to design a new group key computation method for initial and auxiliary key agreement operations [1]. We are more interested in the implementation of group key agreement schemes in ad hoc networks versus contributing toward a new mathematical model.

[1]Group key agreement supports other important security objectives for group communication, such as confidentiality, authentication and nonrepudiation [80] [11]. The properties of contributory group key agreement can be found in [103].

[2]For this reason we conclude in [19] that a topology independent group key agreement scheme may be most suitable for ad hoc networks. However, this means the scheme will rely on broadcasting to distribute contributions which is not ideal.

4.1.1.6 Bootstrapping Security for Dynamic Peer Groups

DPGs have no central point of control and special roles, such as a group leader, are also not fixed *prior* to group formation [1] [79]. DPGs have many-to-many communication patterns and are by nature dynamic in membership, i.e. members may join and leave frequently [1] [103] [79]. The underlying GCS will need to support the dynamic group membership, which have a direct impact on the properties of a suitable GCS. Existing group key agreement schemes for DPGs such as Tree-based Group Diffie-Hellman (TGDH) [9], Cliques [1] and the Burmester and Desmedt *BD* protocol [68] attempt to reduce the impact of auxiliary key agreement operations, but cannot be directly implemented in ad hoc networks [19].

4.1.2 Our Contribution

This chapter significantly expands on our preliminary work [16] [19] [12] [17] [13] considering the given problem statement. We contribute a Group Key Management scheme, called AdHocGKM, to bootstrap the security of a GCS for ad hoc networks [2]. We consider the following GCS protocol architecture, namely 1) unicast routing, 2) group membership services, 3) multicasting, 4) group key agreement and 5) data sharing. We are not aware of any solutions that bootstrap the security of a DPG and the underlying GCS for ad hoc networks.

We also present a novel primary-partition group membership service for ad hoc networks that satisfies the necessary safety and liveness properties for DPGs, based on those defined in [10]. To the best of our knowledge, we are not aware of any group membership service for ad hoc networks that satisfies the necessary safety and liveness specifications, based on those defined in [10], in support of contributory group key agreement. Not considering the impact of group membership services on group key agreement in a critical omission of many existing schemes for ad hoc networks as we discuss in Section 4.2.

As we discuss above, the dynamic network topology and group membership maintenance result in challenging requirements for bootstrapping group communication and security. The main concept underpinning our solution is to exploit the impact of the undeterministic network topology and changing group membership. We will show how we use the effect of unicast and multicast routing failures, and the frequent view maintenance performed by the group membership service, to design a progressively robust scheme.

We evaluate the security and effectiveness of AdHocGKM and show using simulations that the

central key distribution mechanism of proposed scheme results in robust security bootstrapping for group communication with low implementation complexity, suitable for stationary, and low to high mobility ad hoc networks.

4.1.3 Organization of Chapter

The chapter is organized as follows: Section 4.2 briefly reviews related work in the field of group key management with respect to its suitability for mobile ad hoc networks. Section 4.3 provides an overview of AdHocGKM including the system and adversary model (Section 4.3.1), offline initialization phase (Section 4.3.2) and online post-initialization phase (Section 4.3.3). Section 4.4 discusses the security and features of the proposed key management scheme for GCS suitable for ad hoc networking. In summary, we conclude in Section 4.5.

4.2 Related Work

This chapter address group key management taking into consideration peer-to-peer key management [16], group communications systems (GCS) and associated properties [10] [2] [106] [80], multicasting [107], group membership services and the impossibility result [10] [104] [105], unicast routing [108] [109], group key agreement for dynamic peer groups [1] [19] and data replication [110] [111] [112].

We are not aware of any existing literature that consider bootstrapping the security of DPGs and GCS in ad hoc networks similar to what Amir *et al* [80] have proposed for conventional, wired networks [106]. Bootstrapping the security of ad hoc networks is mainly considered in existing literature in terms of peer-to-peer key management [16] [3] [5] [28] and group key agreement closely aligned to schemes for conventional networks [19] [113] [114] [115] [92] [116].

Expanding on our analysis in Chapter 3 and [13] [16] to the fields addressed by the above papers, we briefly discuss the areas that need the most attention in support of reliable group communication, namely group membership services and group key agreement.

Group membership services suitable for ad hoc networks have received limited attention to date and is still an open area for research. The most noteworthy effort is due to Briesemeister *et al* [117] [118], which unfortunately is not suitable for DPGs due to the localization of group membership. The ProbabilistIc Lightweight grOup communication sysTem (Pilot) [2], as the

most prominent GCS for ad hoc networks, mainly focus on partitionable membership services for the multicast protocol, which are not sufficient to support group key agreement as the multicast-group membership service is not designed to maintain a primary-partition view. Furthermore, the membership service of RDG [107] is not based on any explicitly defined safety and liveness properties as defined in [10].

The TransMAN [119] GCS for mobile ad hoc networks is built on a broadcast infrastructure. The approach deployed with this protocol is unlikely to yield a suitable GCS for DPGs and contributory group key agreement given the impossibility result [105] and the dynamic network topology of mobile ad hoc networks.

The slow advance in the area of group key agreement for ad hoc network GCS, in relation to conventional networks, is mainly due to the relatively immature state of GCS for ad hoc networks [120] [10] [80] [2]. We are not aware of any GCS for ad hoc networks that adhere to formal safety and liveness properties for group membership services and multicasting, as defined in [10], that will satisfy the requirements of DPGs and contributory group key agreement. As a result, most existing group key agreement schemes for ad hoc networks [114] [115] [92] [116] [113] [120] assume the existence of a reliable GCS without understanding if the GCS can meet the requirements of group communication for ad hoc networks. The existing group key agreement schemes for ad hoc networks [114] [115] [92] [116] [113] merely adapt schemes for conventional networks [68] [1] [9]. Considering the design challenges and requirements discussed in Section 4.1.1 and our study in Chapter 3, implementing group key agreement needs a more innovative approach that take advantage of the unique characteristics of ad hoc networks by "fighting fire with fire" [121] [2].

4.3 AdHocGKM: Group Key Management in Ad Hoc Networks

The proposed group key management scheme for ad hoc networks, AdHocGKM, bootstraps Group Communication Systems (GCS) and security for group communication by providing keying material to the following components of the GCS [10], namely the unicast routing protocol, membership service, multicast routing, group key agreement and data sharing protocols. Our aim is to design a straightforward and practical group key management scheme that is easy to analyze and implement.

The following section presents the system and adversary model assumed in the remainder of the chapter. Section 4.3.2 explains the offline set up procedure for network participants before they

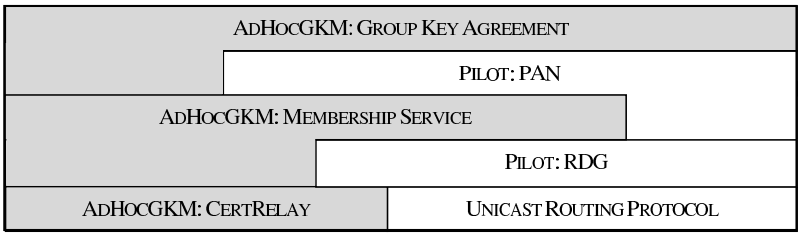

Figure 4.1: Protocol structure of AdHocGKM integrated with Pilot [2]

join the network. In Section 4.3.3 we discuss how AdHocGKM bootstraps security during the post-initialization online phase.

The protocol structure of AdHocGKM integrates with the layered architecture of Pilot [2] as illustrated in Figure 4.1. We highlight the strong dependencies between components of the protocol structure, as emphasized in Section 4.1.1, by showing an overlap on the same layer.

4.3.1 System and Adversary Models

4.3.1.1 System Model

We consider an ad hoc network of wireless nodes with generic medium access control (MAC) and routing mechanisms. The network consisting of tens or hundreds of nodes that move with a random mobility pattern. Nodes can be stationary or move with low to high mobility speeds ($0m/s - 20m/s$). We assume that there are no pre-existing infrastructure, hence the nodes perform all networking functions such as routing. The secure unicast routing protocol is on-demand and based on DSR [108] or AODV [109].

The network has no form of *on-line* trusted authority, hence our scheme does not make use of a distributed certificate authority as proposed in [3] [28]. The scheme requires an off-line trusted authority (TTP) or CA to initialize the nodes prior to joining the network. This assumption is consistent with existing literature [3] [28] [5] and as noted in [28] allows for a high-level of protection to secure high-value communication in, for example, military type applications or in any group which requires strong access control.

We assume AuthBasedPKM as proposed in Chapter 2 as the underlying peer-to-peer key management scheme and will explicitly explain its use in this chapter.

The ad hoc network supports Dynamic Peer Groups (DPGs) [1] in the order of a hundred nodes.

The DPGs make use of a common Group Communication System (GCS). To make the discussion practical we assume the ProbabilistIc Lightweight grOup communication sysTem (PILOT) [2] as the GCS, but note that in principle our solution is not dependent on PILOT specifically. We require the DPG and multicast routing protocol to use the same group membership service. In Section 4.3.3 we discuss the set up of a Storage Set (StS) for PILOT as this is not addressed in [2] [110]. We assume the originator of the DPG to be the group leader (GL). The GL authorize membership requests and maintain the group view.

We use $GES = (M.EG\text{II}.3.\sigma(1), q, s, 1, h(m, q))$ [53] to sign messages, but note it can be replaced by another secure signature scheme with a proof similar to what we present in [12] [13].

We explicitly did not consider distributed-key management and group signatures for this chapter as we present in Chapter 5 and 6 to simplify the discussion. However, we note that DPGs and specifically the StS for PILOT will need such security mechanisms. Distributed-key management and group signatures for ad hoc networks are still open research problems, but we note that the mechanisms presented in this thesis may be used to implement these schemes.

4.3.1.2 Adversary Model

We consider a straightforward, general adversary model, with the adversary model of [28] as a subset. An adversary is a malicious node that uses every means available to break the proposed key management scheme. Any *active* adversary can eavesdrop on all the communication between nodes, modify the content of messages and inject them back into the wireless channel. When a node is compromised all its public and private information is exposed to the adversary. In fact, the Dolev-Yao adversary model [46] is too restrictive, for example, it fails to capture information an adversary may gain from detailed knowledge of the protocols in use. We assume an active, *insider* adversary. The adversary can therefore make use of all the basic network services, such as the routing infrastructure.

4.3.2 Offline Initialization Phase of AdHocGKM

Prior to joining the network, each node P_i, for $(1 \leq i \leq n)$, contacts the off-line trusted authority for the system parameters and an authority-based certificate with a base public/private key pair (x_i, y_i). The offline initialization phase of AdHocGKM is similar to AuthBasedPKM given in Chapter 2.3 and is repeated here for clarity and due to some simplification.

The following system parameters and notations are applicable:

p, q	two large primes, such that $q \mid (p-1)$.
g	generator of the cyclic subgroup of order q in $(Z)_p^*$.
$H(\cdot)$	collision free one-way hash function.
x_P	private key of party P.
y_P	public key of party P, where $y_P = g^{x_P} \ mod \ p$.

The certification information for party P_i is set by the CA as $KI_i = [ID_i \parallel ID_{CA} \parallel CertNo_{CA} \parallel IssueDate_{CA} \parallel ValPeriod_{CA} \parallel AddInfo]$, where ID_{CA} is the identity of the CA, $CertNo_{CA}$ a unique sequence number, $IssueDate_{CA}$ the date of issuing the certificate, $ValPeriod_{CA}$ the validity period and $AddInfo$ some additional extension information.

The CA generates a public/private key pair (x_i, y_i), for node P_i and bind the public key to KI_i and y_i to form a base certificate for P_i. The certificate can be verified using y_{CA}. Note that we can also deploy the Offline Authority-Based Public Key Establishment Scheme presented in Chapter 2.3 to prevent the CA from learning x_i.

After P_i obtains a public/private key pair (x_i, y_i), it uses the base key pair to generate a subordinate public/private key pair (x_i', y_i') as follows [12]:

1. P_i chooses a random number $k_i' \in_R [1, q-1]$ and computes $r_i' = g^{k_i'} \ mod \ p$.
2. P_i computes its new subordinate private key as:

$$x_i' = x_i + H(KI_i \parallel r_i')k_i' \ mod \ q, \tag{4.1}$$

   where KI_i is the original keying information supplied by the off-line TTP and r_i' is P_i's public commitment.

3. Finally P_i computes its corresponding subordinate public key as:

$$y_i' = g^{x_i'} = y_i(r_i')^{H(KI_i \| r_i')} \ mod \ p \tag{4.2}$$

Each node generates a unique identifier ($Address_i$) that is bound to its base public key y_i as follows: $Address_i = H(y_i)$

AdHocGKM requires $Address_i$ to be used as the node's network address or as a fixed part of the address. Note that this requirement places no constraints on the structure of the network

addresses: the entire hash output, $Address_i$, can be used in networks with flat, static addresses or only a part of the output can be used in networks with dynamic addressing.

To obtain an explicitly authentic subordinate key pair, P_i uses its newly obtained subordinate private key x'_i to sign its key information content, KI_i (concatenated with its subordinate public key y'_i and public commitment β'_i) via the modified ElGamal signature scheme presented in [12]. P_i's certificate can then be defined as: $Cert'_i = [KI_i \parallel y'_i \parallel CertNo_i \parallel IssueDate'_i \parallel ValPeriod'_i \parallel \alpha'_i \parallel \beta'_i]$, where (α'_i, β'_i) is the appended signature on $(KI_i \parallel y'_i \parallel CertNo'_i \parallel IssueDate'_i \parallel ValPeriod'_i \parallel \beta'_i)$.

Note that P_i's base key pair (x_i, y_i) is never used for any real communication. Rather, each P_i uses its subordinate key pair (x'_i, y'_i) for securing actual communication.

4.3.3 Online Post-initialization Phase of AdHocGKM

4.3.3.1 AdHocGKM Primary-partition Membership Service: Bootstrapping the GCS

The membership service is central to any view-oriented GCS [80] [10]. The membership service maintains the members' *view* of current active and connected group members [10]. When the membership changes, the primary-partition membership service of each member should install a new view after converging on a single, full membership view [10].

In this section we present a robust, primary-partition membership service for mobile ad hoc networks. The protocol is supported by parts of PILOT as shown in Figure 4.1. We discuss how to integrate the protocol into the Route Driven Gossip (RDG) multicast protocol [107] of PILOT taking into consideration its interdependence with the membership service. The protocol replaces the PILOT multicast-group membership service. The membership service of PILOT [2] is a partitionable group membership service and will therefore not support contributory key agreement for DPGs. Our primary-partition membership service also includes a mechanism to establish the PILOT Storage Set (StS) which is not addressed in [2]. The protocol has two parts, "view construction", and "view query and replication":

View construction protocol:

The group leader (GL) as the originator of the group is the first member of the StS. The GL will appoint StS members from its $View^{gid}$, with unique group ID (gid), depending on the nature of the network. For example, a network with high mobility where users experience poor connectivity

will require more StS members to ensure reliability. Assuming the existence of a GL is reflective of how groups are formed in practice. The GL will most likely be self-appointed or appointed by group communication participants after network formation. The StS also have a unique group ID (StSid). The GL updates $View^{\text{StS}id}$ as the membership of the StS changes.

The GL may invite users directly to join the group (using a JoinInvite message) or flood the network with a periodic or ad hoc advertisement (GroupAdvert). Invited members responding to the JoinInvite, reply to the GL with a JoinAccept or JoinReject. Users responding to a GroupAdvert send the GL a GroupRequest. If the GL authorizes the request, a GroupReply is sent to the new member. After receiving a JoinAccept or sending a GroupReply, GL updates its $View^{gid}$ as described in [2]. GL always append $View^{\text{StS}id}$ to JoinInvite and GroupReply messages. When a member receives $View^{\text{StS}id}$ the routing entry per element of $View^{\text{StS}id}$ is checked and updated as necessary. The Pilot gossip protocol is also used to update $Aview^x$ if a route exists to a node in $Pview^x$ (for $x \in [gid, \text{StS}id]$).

The GL may appoint members of the StS to act as *agents* as required by Pan [2] and also send JoinInvite, GroupAdvert and GroupReply messages. The GroupAdvert messages will be broadcast is such a way to avoid redundancy, that is, appointed StS members will only send GroupAdvert messages if they have not forwarded a GroupAdvert within a specific timeframe given by the GL. StS members update their $View^{gid}$, similar to GL, after receiving a JoinAccept or sending a GroupReply.

All messages are signed by the sender using the generalized ElGamal Scheme: $GES = (M.EG\text{II.3}.\sigma(1), q, s, 1, h(m, q))$ [53] to thwart active adversaries.

The membership service implements the *Self Inclusion* and *Local Monotonicity* properties by forcing members to always include themselves in $View^{gid}$ and by identifying views in increasing order [10].

View query and replication protocol:

Each group member periodically invokes the Probabilistic quorum system for ad hoc networks (Pan) [110], as part of Pilot, to query $View^{gid}$ and $View^{\text{StS}id}$ from the StS. The result is that $View^{gid}$ and $View^{\text{StS}id}$ are continuously replicated between StS members by an enhanced Pan server query protocol and different versions of $View^{gid}$ are merged into a single, full membership view. The periodic queries ensure that a single view is shared across the group, *gid*, even for highly dynamic network topologies with frequent route failures. AdHocGKM therefore exploits Pan to ensures that each group member have a consistent $View^x$. Again, when a member receives new

$View^x$, the routing entry per element of $View^x$ is checked and updated as necessary.

When the membership service detects a view change as defined in [10] it sends a *view_chng* message to the application. All messages from the application will be blocked before the first *view_chng* to enforce the *Initial View Event* property [10]. The frequency of view queries is dictated by the GL in replies to queries. GL monitors changes of $View^{gid}$ and $View^{\text{STS}id}$ as the data objects are replicated by STS members. The frequency of view queries is increased with an increase in view changes for progressive robustness.

The PILOT leave session, based on the gossip protocol are replaced by including the *leaveFlag* in view queries which is used by the STS members to update $View^{gid}$ and $View^{\text{STS}id}$.

4.3.3.2 Bootstrapping the Security of the Unicast Routing Protocol

As the network is formed, nodes have not shared any keying material *a priori* to secure the routing protocol. In order to bootstrap the security of the unicast routing protocol, keying material must be distributed in support of security mechanisms that can provide message integrity and authentication. Our scheme presented in Chapter 2.4.3.1, called Certificate Dissemination based on Message Relaying (CertRelay), propagates certificates along virtual chains via a message relaying mechanism triggered by the routing protocol's control packets [17]. An important objective of CertRelay is to break the routing-security interdependence cycle [33], hence the key distribution scheme does not rely on the routing protocol to provide it with routes.

To make the present chapter self-contained and facilitate subsequent discussions in the context of group communication, we repeat the essence of CertRelay:

While reading the explanation of CertRelay below, it will be useful to keep in mind an existing on-demand routing protocol such as DSR [108] or AODV [109]. Being familiar with the operation of, for example, *endairA* [47], will also help to visualize how the proposed protocol will integrate into a *secure* on-demand routing protocol.

CertRelay, is derived from the following straightforward procedure, illustrated in Figure 2.1:

When a node (RN) receives a routing control packet it checks in its certificate database if it has the certificates of the packet originator (ON) and the previous-hop node (PN) on the forward route. If RN has both the certificates of ON and PN ($Cert_{ON}$ and $Cert_{PN}$), it can process the control packet as normal. If not, it requests both the certificates from PN. If RN does not have the

certificate of PN it also sends its own certificate with the request to the previous-hop. Note that if RN is the first-hop on the route, then the previous-hop node and the control packet originator node will be the same entity. The routing messages thus effectively chain nodes together and allow them to relay all keying material, as required, along the virtual chains.

Table 2.1 explains CertRelay's core procedure in more detail from the routing control packet (RCP) receiver node's perspective (see Figure 2.1).

4.3.3.3 Bootstrapping the Security of the Membership Service, Multicasting and Data Sharing Protocols

CertRelay provides group members of $View^{gid}$ and $View^{\text{STS}id}$ with the necessary keying material to secure the membership service messages in the *view construction* phase. As the unicast routing protocol set up routes to deliver these messages, CertRelay exploits the routing protocol to exchange the required keying material. We avoid the interdependence between the membership service and multicast protocol during bootstrapping of the GCS by not using multicasting to disseminate any of the view construction messages and do not replicate $View^{gid}$ and $View^{\text{STS}id}$ between STS members.

The *view query and replication* phase depends on the RDG multicast [107] and unicast routing protocol for message delivery. In turn, RDG disseminates packets stored in *Buffer* to a fanout F of other group members, randomly chosen from $Aview^{x}$ [2]. RDG depends on the unicast routing protocol to distribute packets and update the routing information associated with $Aview^{gid}$ or $Aview^{\text{STS}id}$. As a result, considering the operation of CertRelay, the necessary keying material is distributed automatically by CertRelay to bootstrap the security of the RDG protocol for all entries in $Aview^{x}$. This is a good case in point of the usefulness of CertRelay to inherently bootstrap the security of the PILOT GCS.

The security of the PILOT data sharing protocol, PAN, is also bootstrapped by CertRelay as keying material is delivered to the unicast routing, membership service and RDG protocols and needs no further explanation.

4.3.3.4 Bootstrapping the Security of the Group Key Agreement Protocol

We consider the *BD* key agreement protocol [68] (see Chapter 3.9) to show how AdHocGKM bootstraps the security of the group key agreement protocol, and how to implement such schemes

in ad hoc networks. We address the practical implementation of *initial key agreement* (IKA) and *auxiliary key agreement* (AKA) operations, as defined by Steiner *et al* [1].

Note that our focus is on the implementation of group key agreement in ad hoc networks and not to design a new group key computation method or mathematical model (see Section 4.1.1). We could also have chosen, for example, Tree-based Group Diffie-Hellman (TGDH) [9] or Cliques [1]. However, based on the outcome of our studies in Chapter 3, the BD protocol may the best suited.

The following additional notation is defined for use in the following text:

n Total number of protocol participants.

i, j, k, l Indices of group members $(i, j, k) \in [1, n]$.

P_i $i-th$ Group member.

K_n Shared group key between n members.

K'_n Updated shared group key between n members, after auxiliary key agreement.

BD initial key agreement (IKA):

The IKA protocol allows group members P_i, $i = [1, n]$ with $View^{gid}$ obtained from the StS to establish a contributory group key K_n over an insecure channel. The resulting group key K_n is secure against *active* adversaries within the given system and adversary model (Section 4.3.1).

- *IKA Protocol: Round* 1

The IKA protocol is initiated by the GL who disseminates a GroupSecureIKA message, using RDG [2], with the $View^{gid}$ data object version included in the message. Each member P_i that receives GroupSecureIKA checks the $View^{gid}$, computes a random secret r_i and invokes the Pan server update protocol to upload $z_i = \alpha^{r_i}$ to the StS.

Each P_i uses the generalized ElGamal Scheme: $GES = (M.EG\text{II}.3.\sigma(1), q, s, 1, h(m, q))$ [53] to generate a signature (q_i, s_i) on z_i which is also uploaded. The signatures thwart the active adversary from modifying keying material.

As z_i gets replicated within the StS, members of the StS will verify the integrity of the data object by validating the signature. With reference to our discussion on bootstrapping the security of GCS in Section 4.3.3.3, it should be clear that CertRelay will distribute the certificate of P_i as necessary.

Upon receiving GROUPSECUREIKA each STS and P_i start a *gather* countdown timer τ_{g1} with a value given by GL. On timeout of τ_{g1}, P_i starts a *commit* countdown timer τ_{c1}. On timeout of τ_{g1}, STS members will not accept any more blinded secrets, z_x, as the gather state is over. Use of the countdown timers for the gather and commit phases follow from Amir [111].

- *IKA Protocol: Round* 2

On timeout of τ_{c1} all participants start a second gather timer, τ_{g2} and P_i queries the STS for z_{i+1} and z_{i-1}. CertRelay will ensure that P_i obtains the necessary certificates to validate the signatures.

The STS agent replies to P_i with the data objects and P_i computes X_i after verifying the signatures as:

$$X_i = (\frac{z_{i+1}}{z_{i-1}})^{r_i} \tag{4.3}$$

It is possible for z_{i+1} and z_{i-1} not to be available as per the agreed $View^{gid}$ version. In this case the STS agent will reply with a GROUPSECUREABORT message. The GL will then restart the protocol with a higher Reliability Degree (R_d) for PILOT [2] or by increasing the STS membership. The GL may choose to exclude the failed representatives from the protocol or this may be applied by default.

- *IKA Protocol: Round* 3

Each P_i uploads X_i to the STS, which will accept the updates until timeout of $\tau g2$. On timeout of $\tau g2$, P_i starts a second commit timer τ_{c2}.

Each P_i uses the generalized ElGamal Scheme: $GES = (M.EG\text{II}.3.\sigma(1), q, s, 1, h(m, q))$ [53] to generate a signature (q_i, s_i) on X_i which is also uploaded to STS.

- *IKA Protocol: Round* 4

On timeout of τ_{c2}, P_i query the STS for $X_k \mid k \in [1, n] \forall k \neq i$. CertRelay will deliver the required certificates to P_i.

The STS agent replies to P_i with the data objects and P_i computes the group key after verifying the signatures:

$$K_{gid} = z_{i-1}^{n \cdot r_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \ldots \ X_{i+2} \ mod \ p \tag{4.4}$$

It is possible for all the required X_k data objects not to be available as per the agreed $View^{gid}$

version. Again, the StS agent will reply with a GroupSecureAbort message. The GL will then restart the protocol with a higher Reliability Degree (R_d) for PILOT [2] or by increasing the StS membership. The GL may choose to exclude the failed representatives from the protocol or this may be applied by default.

BD auxiliary key agreement (AKA) - membership addition (MA):

The AKA operations under the BD protocol are conveniently very similar. We therefore only consider Membership Addition (MA) as the principles also apply to Mass Join, Membership Exclusion, Mass Leave, Group Merge, Group Partition and Key Refresh operations as defined in [1].

In order to ensure *key freshness*, at least one of the existing group members (P_k) has to renew its share in the group key K_n by generating and distributing a new blinded secret z'_k for all AKA operations.

With the membership addition protocol a new member P_j is added to the existing group. The group members, including the new member, establish a updated group key K'_n over an insecure channel. The new group key K'_n is secure against *active* adversaries.

- AKA MA Protocol: Round 1

P_j computes a random secret r_j and includes $z_j = \alpha^{r_j}$ with GroupRequest or JoinAccept. When a member of StS, P_i, receives a JoinAccept or send a GroupReply, the AKA MA protocol is initiated by the GL with the dissemination of a GroupSecureAKA message with $View'^{gid}$ using RDG.

Group member P_i also computes a new random secret r'_i and $z'_i = \alpha^{r'_i}$ to ensure key freshness. Blinded secrets z'_i and z_j are signed with $GES = (M.EG\text{II}.3.\sigma(1), q, s, 1, h(m, q))$ [53].

All members that receive GroupSecureAKA send a *block* message to the application before installing the new $View'^{gid}$ and start gather countdown timer τ_{g3}. Blocking the application from sending messages for a certain period before installing the new view ensure *Sending View Delivery*, which is important for view members to encrypt and decrypt data in the same view [10].

Member P_{i-1} queries the StS for z'_i, P_{i+1} queries the StS for z_j and P_j queries the StS for z'_i and z_{i+1}. CertRelay will distribute the certificate of P_i as necessary.

When the StS agents respond with the data objects, P_{i-1}, P_i, P_j and P_{i+1} validate the signatures and compute the following shares respectively:

$X_{i-1} = (\frac{z_i}{z_{i-2}})^{r_{i-1}}$, $X_i = (\frac{z_j}{z_{i-1}})^{r'_i}$, $X_j = (\frac{z_{i+1}}{z'_i})^{r_j}$ and $X_{i+1} = (\frac{z_{i+2}}{z_j})^{r_{i+1}}$.

P_j therefore takes the ordered position of P_{i+1}, and P_{i+1} will become P_{i+2},...., and P_n become P_{n+1}.

The protocol will respond similar to the IKA protocol by sending a GroupSecureAbort message, increasing robustness and restarting the protocol should the blinded secrets not be available.

- AKA MA Protocol: Round 2

Each P_k, $k \in [i-1, i, j, i+1]$ uploads X_k to the StS, which will accept the updates until timeout of τ_{g3}. On timeout of τ_{g3}, P_i starts a new commit timer, τ_{c3}.

Each X_k is signed with $GES = (M.EG\text{II}.3.\sigma(1), q, s, 1, h(m, q))$ [53].

- AKA MA Protocol: Round 3

On timeout of τ_{c3}, P_k queries the StS for $X_l \mid l \in [i-1, i, j, i+1] \forall l \neq k$. CertRelay will relay the certificates as necessary.

The StS agent replies to P_k with the data objects and P_k computes the group key after verifying the signatures as:

$$K'_{gid} = z_{k-1}^{n \cdot r_k} \cdot X_k^{n-1} \cdot X_{k+1}^{n-2} \ldots X_{k+2} \; mod \; p \tag{4.5}$$

The protocol will respond by sending a GroupSecureAbort message, increasing robustness and restarting the protocol should the intermediate keying material not be available to compute the group key.

4.4 Discussion on the Features and Security of AdHocGKM

This section discusses the features and performance of the proposed scheme within the context of the problem statement given in Section 4.1.1.

4.4.1 On the Features of AdHocGKM

As we explained in Section 4.1.1, the main challenge to bootstrapping the security of group communication for ad hoc networks is the combined impact of the dynamic network topology and

frequent changes in group membership. As noted in [2], changes in membership require that the membership service constantly update the group *view*, while sporadic connectivity has a similar effect to rearranging the partitionable view as corresponding route entries become invalid. Route failures make it difficult, if not impossible, to establish a single, full membership view using only conventional unicast or multicast routing protocols. AdHocGKM is specifically designed to eliminate the impact of the dynamic and unpredictable network topology (as the cause of route failures) on the membership service and group key agreement scheme.

4.4.1.1 Progressively Robust Key Distribution

The first mechanism deployed by AdHocGKM to overcome the dynamic network topology is to exploit the inherent ability of the routing protocol to recover from route failures. CertRelay propagates keying material along virtual chains via a certificate relaying mechanism triggered by the routing protocol control messages. The protocol does not need a route to relay certificates and therefore breaks the routing-security interdependence cycle. As the on-demand routing protocol maintains routing tables, CertRelay distributes only the minimum key material necessary to secure the unicast routing protocol. Nevertheless, any group communication protocol, such as the membership service, multicasting and data sharing protocols that are dependent on the underlying unicast routing protocol will indirectly fuel the rate at which CertRelay set up security associations as the routing protocol is activated (We confirm this observation with simulation results in Section 2.5.2.3.

The usefulness of CertRelay in the context of bootstrapping the security of group communication is demonstrated in Section 4.3.3.3 and Section 4.3.3.4, as we show how CertRelay distributes all necessary keying material (certificates) in support of the GCS.

In Section 4.4.2 and Section 4.4.3 we discuss the security and performance of CertRelay.

4.4.1.2 Progressively Robust Group Membership Service

The second mechanism deployed by AdHocGKM to overcome the dynamic and unpredictable network topology and mitigate the impact on membership changes, is to exploit the two-layer Pilot GCS [2] to implement robust group membership service and key agreement. As shown in Figure 4.1, AdHocGKM integrates with Pilot's RDG and Pan protocols [107] [110]. The figure also shows the interdependency between the components of the protocol architecture as discussed

in Section 4.1.1.

To take account of how AdHocGKM exploits RDG and PAN, similar to how CertRelay exploits the unicast routing protocol, it is important to understand the properties and operation of PILOT. RDG is designed to overcome the undeterministic nature of ad hoc networks by taking a probabilistic approach to multicasting. Multicast packets are distributed by gossiper-push and gossiper-pull mechanisms [107]. PAN implements a probabilistic quorum system on a subset of network nodes called the Storage Set (STS). PAN uses RDG to diffuse data object updates to a random set (write quorum) and to access a read quorum in STS. The STS ensures that the proposed membership service adhere to the *Primary Component Membership* property [10].

We highlight key features of how AdHocGKM constructs and maintains the $View^x$ using RDG and PAN:

View construction protocol:

AdHocGKM bootstraps the GCS by implementing the join session using only unicast routing and therefore obtains initial keying material from CertRelay. To eliminate a single point of failure the Group Leader (GL) shares the responsibility of inviting new members and advertising the group with other STS members. The GL and STS members bootstrap new group members by sharing $View^{\text{STS}id}$ and acting as *agents* for PAN. The leave session is implemented by including the *leaveFlag* in only view queries which is used by STS members to update $View^{gid}$ and $View^{\text{STS}id}$.

View query and replication protocol:

When group members in $View^{gid}$ query the latest view from STS, by accessing a read quorum, the STS members replicate the view and merge views into a single, full membership view. This is a key feature that, combined with the view construction method, allows for a primary-partition membership service. We therefore exploit the inherent capability of the RDG and PAN protocols to "*fight fire with fire*"[3] to implement a primary-partition membership service contrary to common belief that this is not practical in ad hoc networks [2] [117] [118]. AdHocGKM exploits the PAN server query protocol to maintain a consistent group *view* as RDG continuously replicate the latest view between STS members as the view is queried by group members. In fact, the more dynamic the network topology and the more frequent the membership changes the better our membership service performs; GL will increase the frequency of view queries as more frequent changes in $View^{gid}$ and $View^{\text{STS}id}$ are detected which in turn increase the replication of the group

[3]This expression was used for gossip-based protocols [121] [2] to address different problems in the area of multicasting.

view by PAN between StS members.

4.4.1.3 Progressively Robust Group Key Agreement

AdHocGKM leverage the *BD* group key mathematical model [68] to demonstrate a method for implementing robust group key agreement in ad hoc networks based on the PILOT [2] GCS. This represents the third mechanism deployed by AdHocGKM to overcome the dynamic and unpredictable network topology. We use the probabilistic quorum system implemented by PAN [110] to reliably exchange group key shares during IKA and AKA operations.

To ensure that the group key is representative and the protocol overcomes delays in updating the StS, AdHocGKM implements countdown timers to allow for *gather* and *commit* states, similar to what Amir [111] proposed for Spread [106] [80]. At the end of the gather phases the StS may detect that it is not possible to move to the next round since all the group key shares (z_{i-1}, z_{i+1} or X_i) are not available and therefore abort the protocol. Consequently, the GL increases the reliability degrees of RDG and PAN or increase the size of the StS before restarting the protocol. Again, the more dynamic the network topology, AdHocGKM responds by increasing robustness progressively. The robustness of the protocol can also be improved by excluding failed representatives from subsequent protocol runs and the membership addition protocol can be invoked to include them later.

4.4.2 On the Security of AdHocGKM

Considering the proposed scheme's system and adversary models given in Section 4.3.1 and the protocol overview in Section 4.3, we discuss the security of CertRelay, group membership service and key agreement scheme. Overall we attempt to take a sensible approach to arguing the security of AdHocGKM [49].

We are not concerned with the security of the RDG and PAN protocols of PILOT [2] and assume these protocols to be secure. We also assume the unicast routing protocol to be secure [47].

4.4.2.1 On the Security of CertRelay

We discuss the security of CertRelay in sufficient detail in Chapter 2.5.1.3. The security of CertRelay is important for AdHocGKM since CertRelay bootstraps the security of all the GCS protocols.

4.4.2.2 On the Security of the AdHocGKM group membership service

The membership service *view construction* protocol leverage of the keying material provided by CertRelay to secure JoinInvite, JoinAccept, JoinReject, GroupAdvert, GroupRequest and GroupReply messages. The integrity of messages is protected by the generalized ElGamal Scheme: $GES = (M.EG\text{II}.3.\sigma(1), q, s, 1, h(m, q))$ [53] or a similar scheme. It is possible to prove the security of this scheme, similar to the approach we followed in [12] [13], in the Random Oracle and Generic Model (ROM+GM) proposed by Schnorr *et al.* [51] [52].

The membership service *view query and replication* protocol fully relies on the security of the underlying unicast protocol and PILOT RDG and Pan protocols which we assume to be secure.

4.4.2.3 On the Security of the BD group key agreement protocol

We consider the security of the AdHocGKM BD IKA and AKA group key agreement protocols.

IKA operation:

In [122], Katz *et al.* give a *full* security proof of the IKA BD protocol, in a precise and widely excepted model. This proof will not be repeated here and we cite [122] for details. In order to argue the applicability of this security proof to AdHocGKM's BD IKA protocol, the major differences between the BD protocol [68] and the AdHocGKM BD IKA protocol (Section 4.3.3.4) are identified:

- AdHocGKM is an *authenticated* key agreement protocol. This implies that it can withstand attacks from *active adversaries* in contrast to the BD protocol, which is only secure against *passive* adversaries[4].
- AdHocGKM takes into consideration that n *simultaneous* broadcasts is impractical in ad hoc networks. It uses the PILOT StS as a reliable intermediary between group members to ensure predictable and consistent exchange of group key shares.

The authentication mechanism of AdHocGKM is consistent with the guidelines provided by the *compiler* presented in [122], which converts any group key agreement protocol into an *authenticated* group key agreement protocol. It is also pointed out in [123] that converting from n simultaneous

[4]Note that the *authentication mechanism* of AdHocGKM can be enhanced by using *nonces* and *sequence numbers* to guarantee the *freshness* of messages and to prevent undetectable replay [11]. For the sake of simplicity these enhancements were not included in the protocol description.

broadcasts in a single round, to n rounds, each with a single broadcast, does not affect the notion of security in the given model. We can also argue that our communication convention is irrelevant to our notions of security.

Katz *et al.* [122] do not consider queries to the *Corrupt oracle*, since the BD protocol does not make use of long term keys. Due to the properties of the AdHocGKM *subordinate key pair* generation procedure (Section 4.3.2) it can be shown that the *Corrupt oracle* can be ignored in the case of AdHocGKM. The *subordinate key pair* generation procedure also contributes to ensuring perfect forward secrecy (PFS) [103].

Based on the discussion it can be concluded that the security proof of the BD protocol presented in [122], is applicable to the BD IKA protocol implementation of AdHocGKM.

AKA operation:

The security for the AdHocGKM AKA operations can be proved using similar arguments as used by Steiner *et al.* [1].

Proof. Assume that the following sets are defined: C, the set of all *current* group members, P, the set of all *past* group members and F, the set of all future group members. It has to be shown that P and F is unable to compute the current group key:

$$K'_n = g^{r_1 r_2 + \cdots + r_{j-1} r'_j + r'_j r'_{j+1} + \cdots + r_n r_1}$$

Following the proof in [1], all members P and F are collapsed into one *super* adversary Eve such that $Eve = P \cup F$. Assume that member P_j is a founding member and is responsible for updating its share during *all* AKA operation. Since Eve knows $r_i | P_i \in Eve$, the group key can be written as:

$$K'_n = g^{B[\sum (E)]},$$

where B is a constant known to Eve and $E = \{r_1 r_2, \ldots, r_{j-1} r'_j, r'_j r'_{j+1}, \ldots, r_n r_1\}$ are the $\binom{n}{2}$ secret key pairs of C.

From the the procedure used to computer the random secret r_j it should be clear that there is no dependance between the secret shares r_j, r'_j and r''_j of member P_j.

In the view of the adversary the only values that contain r_j is the values X_{j-1}, X_j and X_{j+1}

broadcasted by P_{j-1}, P_j and P_{j+1} respectively. Taking a closer look at

$$X_{j-1} = (\frac{z_j}{z_{j-2}})^{r_{j-1}} = g^{(r_j - r_{j-2})r_{j-1}},$$

it is noted that X_{j-1} is also independent from X'_{j-1} where:

$$X'_{j-1} = (\frac{z'_j}{z_{j-2}})^{r_{j-1}} = g^{(r'_j - r_{j-2})r_{j-1}}$$

It is in the view of the author that P_{j-1} will not leak any information about its random secret r_{j-1} to *Eve*, since z_{j-1} will also be independent from z'_{j-1} and so forth. The same applies to the private key r_{j+1} of P_{j+1}. Leaking information can be further prevented by periodic key refresh procedures, which are in either case required to limit the information available to adversaries for cryptanalysis [11].

It is also assumed that *Eve* knows all:

$$\{g^{\sum S} | S \subset E\}$$

If $view(n, S)$ is defined as the ordered set of all proper subsets of $\{r_1 r_2, \ldots, r_{j-1} r'_j, r'_j r'_{j+1}, \ldots, r_n r_1\}$, then it follows from the proof by Katz *et al.* [122] that $(view(n, S), K_n)$ is *polynomial indistinguishable* from $(view(n,S), y)$ for a randomly chosen value $y \in G$, where G is a cyclic subgroup of order q in $(Z)^*_p$. From the definition of *view* by Steiner *et al.* [1], *Eve*'s knowledge is a subset $view(n, E)$.

Substituting S with E, it follows that $(view(n, E), K'_n)$ is *polynomial indistinguishable* from $(view(n, E), y')$ for a randomly chosen value $y' \in G$.

It can thus be concluded that the AKA operations of AdHocGKM enjoy the same *strong* security properties in the *standard model* [122], as the AdHocGKM IKA operation. □

4.4.3 On the Performance of AdHocGKM

As shown in Figure 4.1, the AdHocGKM protocol structure includes three protocols, namely CertRelay, group membership service and group key agreement. The performance of the latter

two protocols are directly dependent on the performance of the PILOT RDG and PAN protocols. As shown through analysis and simulation in [2] [107], RDG is a reliable and scalable multicast protocol. The predictability and reliability of PAN is also confirmed through analysis and simulation in [2] [110]. A similar analysis was performed by Luo *et al* for DICTATE [28].

Simulations is a standard way in ad hoc networks to study the performance of protocols due to the complexities introduced by, for example, the wireless channel, dynamic peer-to-peer routing and node mobility. Currently the commonly used ns-2 simulator [61] does not have a standard implementation of a GCS for ad hoc networks which makes benchmarking of new group communication protocols with ns-2 difficult. The performance of our membership services and group key agreement are also dependent on many external factors such as the frequency of membership changes, RDG and PAN parameters (F, R_ds, R_dc , τ_q, $\hat{\xi}_W$, $\hat{\xi}_R$ etc.) [2], group communication traffic patterns and size of the STS [2]. An experimental testbed implementation, as used for testing Secure Spread [80], is not practical for mobile ad hoc networks as it is difficult to recreate test runs of a dynamic network topology. Fortunately, arguing the essence of our membership service and group key agreement protocols within the given problem statement can be based on the simulation results in [2] [107] [110] [28] and the discussions in Section 4.4.1 and Section 4.4.2.

Nevertheless, CertRelay as proposed in Chapter 2 is at the center of bootstrapping security for all the GCS protocols discussed in this chapter. The performance of CertRelay is addressed in Chapter 2.5.2 based our extensive simulations. We concluded that the scheme's communication and computational overhead has negligible impact on network performance. The results in Chapter 2.5.2 are directly applicable to AdHocGKM and will not be repeated here.

4.5 Conclusions

This chapter identifies two related problems, namely bootstrapping Group Communication Systems (GCS) for Dynamic Peer Groups (DPGs) in ad hoc networks and bootstrapping the security of the underlying protocols. These protocols include unicast routing, multicasting, group key agreement and data sharing. In response to the problems, we contribute a robust primary-partition membership service integrated with our group key management scheme called, AdHocGKM. To make the discussion practical we show how AdHocGKM will work with the noteworthy PILOT GCS for ad hoc networks proposed by Luo *et al* [2].

AdHocGKM exploits the combined impact of the dynamic network topology and frequent changes

in group membership to "fight fire with fire" [121] [2]. The performance of the scheme improves as the network topology and group membership becomes more dynamic. To achieve this objective, AdHocGKM deploys multiple, innovative mechanisms at the various layers of the GCS protocol architecture to improve reliability and security as the environment becomes increasingly hostile.

- PROGRESSIVELY ROBUST KEY DISTRIBUTION: On the unicast routing layer our key distribution scheme, called CertRelay, exploits the inherent ability of the routing protocol to recover from route failures. CertRelay propagates keying material along virtual chains via a certificate relaying mechanism triggered by the routing protocol control messages. The protocol does not need a route to relay certificates and therefore breaks the routing-security interdependence cycle. The usefulness of CertRelay is demonstrated by its ability to bootstrap the security of all the protocols in the GCS architecture.

- PROGRESSIVELY ROBUST PRIMARY-PARTITION MEMBERSHIP SERVICE: The second mechanism deployed by AdHocGKM is to exploit the two-layer PILOT system [2] to implement a robust group membership service. Similar to how CertRelay exploits the unicast routing protocol, the proposed group membership service exploits the inherent capability of PILOT to implement progressive robustness. AdHocGKM uses the data sharing service to maintain a consistent group *view*; as the view is periodically queried by group members the multicast protocol continuously replicate the latest group *view* between members of the data storage set (STS). The group leader monitors changes in the view and increase the frequency of view queries by group members to counter the impact of an increase in membership changes on the reliability of the membership service. The group membership service also bootstraps the STS which is a shortcoming of PILOT.

- PROGRESSIVELY ROBUST CONTRIBUTORY GROUP KEY AGREEMENT: AdHocGKM overcomes the dynamic and unpredictable network topology by using the probabilistic quorum systems implemented by PAN [110] to reliably exchange group key shares during IKA and AKA operations. In the event of protocol failures the scheme responds by excluding failed representatives and increasing the robustness of the PILOT multicasting and data sharing protocols. The redundance introduced by the STS enable group members to be included in the contributory group key in a predictable and controllable fashion. In fact, we claim that our approach supported by PILOT makes contributory group key agreement possible in ad hoc networks with highly dynamic and unpredictable topology changes.

The simplicity of AdHocGKM allows for strong security arguments for its various components, as discussed above, in a widely accepted adversary model. The GCS protocols use the underlying

key distribution scheme, CertRelay, to distribute only authenticated information on a peer-to-peer basis, which provides for robust operation and provable protection against forgery and undetected modification.

The AdHocGKM CertRelay protocol, is shown to be at the center of bootstrapping the security of the GCS and our main concern from a performance perspective. The effectiveness of CertRelay, its low implementation complexity and ease of integration with existing secure routing protocols were verified through coding and simulating the scheme in ns-2. The results presented in Chapter 2.5.2 show that AdHocGKM has negligible impact on the network performance. It was concluded that the message relay mechanism provides an efficient way to manage keying material. The performance of AdHocGKM improves as the network topology becomes more unpredictable and dynamic, showing that we can make security progressively robust in a hostile environment.

Part III

Distributed-Key Management in Distributed Communication Systems

Chapter 5

Distributed-Key Management in Distributed Communication Systems

5.1 Introduction

In distributed systems it is sometimes necessary for users to share the power to use a cryptosystem [124] [125]. The system secret is divided up into shares and securely stored by the entities forming the distributed cryptosystem. For example, in many applications a threshold (t) or more shareholders are required to cooperatively generate a digital signature in contrast to the conventional single signer. This may also be seen as a distribution of trust (or power) since the shareholders must collaborate and contribute equally to produce a valid *multiparty* signature. The main advantage of a distributed cryptosystem is that the secret is never computed, reconstructed, or stored in a single location, making the secret more difficult to compromise [126]. It is generally assumed that an adversary will find it difficult to corrupt t honest parties needed to reconstruct the secret key.

An important component of any threshold digital signature scheme is the sharing of the group key [125]. If the group key sharing does not involve a trusted third party (TTP), the key sharing is performed by a distributed collaborative protocol. A threshold cryptosystem that is suitable

for distributed systems (for example ad hoc networks [3]) may require a *distributed-key generation* (DKG) protocol that effectively eliminates the TTP. Removal of an online TTP without a priori share distribution by an offline TPP is the most difficult to achieve.

The first DKG protocol, for discrete logarithm based threshold cryptosystems, was introduced in [14] and a number of promising proposals for DKG have been published since [127] [126] [128] [129] [130] [131]. In these schemes the distributed-key is generated in a distributed fashion as a function of the random input of all protocol participants while preserving the symmetric relationship between them.

In threshold cryptosystems it is impractical to assume that an adversary cannot compromise more than t shareholders during the entire lifetime of the distributed secret [132]. The secret shares therefore have to be periodically updated using a *distributed-key updating* (DKU) protocol. The DKU scheme allows for only an "impractical" period T in which an active/mobile adversary has time to compromise a sufficient percentage of the group secret shares [132]. Many of the existing DKG schemes ([14] [127] [126] [128] [129] [130] [131]) do not consider the problem of DKU.

In addition to periodically updating the secret shared, the access structure $\Gamma_P^{(n,t)}$ of the initial share distribution will not necessarily remain constant [133]. Assuming the same shareholders to be present at all times is unrealistic and the availability/security tradeoff (set by the total number of group members (n) and threshold (t)) may need to be changed as a function of system vulnerability, networking environment and current functionality of the cryptosystem. The shares must then be redistributed to a new access structure $\Gamma_{P'}^{(n',t')}$ using a *distributed-key redistribution* (DKR) protocol [133] [134]. Many of the existing DKG schemes ([14] [127] [126] [130] [131]) also do not consider the problem of DKR.

In [18], we define a protocol suite capable of servicing *all* aspects of secret sharing and define this as a ***distributed-key*** *management infrastructure* (DKMI). The definition of DKMI thus includes DKG, DKU and DKR.

The main objective of this chapter is to propose a DKMI with specific focus on combining DKG, DKU and DKR into a single protocol. The proposed DKG protocol is based on the scheme due to Zhang *et al.* [131]. Using the proposed DKG protocol as a basis we significantly extend the field a distributed-key management by solving problems with and simplifying existing DKU and DKR schemes. We therefore effectively combine three distinct fields into one and argue that these fields cannot be considered in isolation as an adversary may compromise the threshold cryptosystem by attacking the DKG, DKU or DKR protocols.

The DKMI presented in this chapter is a critical building block for the threshold multisignature scheme [15] proposed in the next chapter.

The chapter is organized as follows: In Section 5.2.2 we present a modified version of the Zhang *et al.* [131] DKG protocol. Section 5.3 introduces a new distributed-key redistribution/updating (DKRU) scheme which combines DKU and DKR into a single protocol. The DKG protocol are aligned to the DKRU protocol to form a single protocol in support of the ***distributed-key*** *management infrastructure* (DKMI). The security and features of the proposed DKMI are discussed in Section 5.4. Some conclusions are provided in Section 5.5.

5.2 Proposed Distributed-key Management Infrastructure

In this section we propose a complete ***distributed-key*** *management infrastructure* (DKMI), without assistance from a trusted key distribution center (KDC) or online authority.

5.2.1 System model

The system model and assumptions defined in Zhang *et al.* [131] applies.

The group members P_i, for $i = 1 : n$ agree on and publish the following system parameters:

p, q two large primes, such that $q \mid (p-1)$.

g generator of the cyclic subgroup of order q in $(Z)_p^*$.

$H(\cdot)$ collision free one-way hash function.

(n, t) threshold parameter t and total number of group members n.

T threshold cryptosystem secret update period.

Protocol participants P_i, for $i = 1 : n$ are assumed to have a long-term public/private key pair PK_i/SK_i and an authentic certificate, verifiable with the public key of a common trusted third party (System parameters may also be distributed by the authority). The certificate of P_i binds the public key $PK_i = g^{SK_i}$ to user P_i's identity ID_i. The certificates are distributed to (or can be traced by) all communication entities P_j, for $j = 1 : n$, $j \neq i$.

The trusted third party (TTP) is assumed to be offline and therefore not present during formation of the threshold cryptosystem. Group members may be a priori unknown or authorized by the TTP during offline initiation. The method used to control group membership (adding and removing members) is application dependent, but may, for example, be the responsibility of the group chair or TTP.

DKG protocols typically make use of either the Shamir Secret Sharing [135], Feldman Verifiable Secret Sharing (VSS) [136] or Pedersen VSS [14] protocols as a foundation [129]. We will make use of the Shamir scheme.

5.2.2 Modified Zhang *et al.* Publicly Verifiable Distributed-Key Generation Protocol

In this subsection a publicly verifiable *distributed-key generation* (DKG) protocol is presented. The round optimal DKG protocol is developed from the scheme of Zhang *et al.* [131]. The purpose of the protocol is to realize secure, *initial* secret sharing to an access structure $\Gamma_P^{n,t}$, without a trusted dealer or KDC.

The protocol executes in four steps:

1. For $1 \leq i \leq n$, P_i chooses a random number $d_i \in [1, q-1]$. P_i then shares the random value d_i with all the other protocol participants using Shamir's secret sharing scheme [135] as follows:

   (a) Sets $a_{i,0} = s_{i,0} = d_i$ and chooses at random $a_{i,k} \in [1, q-1]$, for $1 \leq k \leq t-1$, such that the numbers $a_{i,0}, \ldots, a_{i,t-1}$ define a polynomial $f_i(X) = \sum_{k=0}^{t-1} a_{i,k} X^k$ over $\mathbb{Z}_q$ of degree $t-1$.

   (b) For $j = 0, \ldots, t-1$, P_i computes, $A_{i,j} = g^{a_{i,j}} \ mod \ p$. For $j = 1, \ldots, n$, P_i computes $s_{i,j} = f_i(ID_j) \ mod \ q$, $y_{i,j} = g^{s_{i,j}} \ mod \ p$ and an encryption $E_{PK_j}(s_{i,j})$ of each secret shadow. P_i then binds itself to the public values by generating the digital signature (r_{P_i}, s_{P_i}) on message $([y_{i,1}, E_{PK_1}(s_{i,1})] \parallel \cdots \parallel [y_{i,n}, E_{PK_n}(s_{i,n})] \parallel A_{i,0} \parallel \cdots \parallel A_{i,(t-1)})$ using the ElGamal type signature variant: $GES = (M.EG\text{II}.3.\sigma(1), r, s, \ h(m,r), 1)$, developed from the generalized ElGamal scheme presented in [53]. The signature is generated with P_i's long-term private key SK_i. P_i broadcasts the message and the signature to all protocol participants. The encryption $E_{PK_j}(s_{i,j})$ of the secret shadow for protocol participant P_j is performed using an appropriate *publicly verifiable encryption*

scheme [127] [137]. P_i keeps the share when $j = i$.

2. Protocol participants $P_1, \ldots, P_n$ each verify the following:

   (a) (r_{P_i}, s_{P_i}) is a valid signature on the public values $([y_{i,1}, E_{PK_1}(s_{i,1})] \parallel \cdots \parallel [y_{i,n}, E_{PK_n}(s_{i,n})] \parallel A_{i,0} \parallel \cdots \parallel A_{i,(t-1)})$ received from P_i. This serves as a proof of integrity and binds participant P_i to these public values.

   (b) $E_{PK_j}(s_{i,j})$ is the correct encryption of $s_{i,j}$ under the public key PK_j. This implies that anybody can verify that $y_{i,j}$ and $s_{i,j}$ satisfy the following relationship: $y_{i,j} = g^{s_{i,j}} \ mod \ p$ and that protocol participant P_j can correctly retrieve the subshare $s_{i,j}$ (Note that only P_j can decrypt $E_{PK_j}(s_{i,j})$ with its corresponding private key SK_j to recover $s_{i,j}$).

   (c) P_j knows that the *subshare* distribution from member P_i is correct if Eq.(5.1) holds.

$$\prod_{k=0}^{t-1} A_{i,k}^{(ID_j)^k} = \prod_{k=0}^{t-1} (g^{a_{i,k}})^{(ID_j)^k} = g^{\sum_{k=0}^{t-1}(a_{i,k})(ID_j)^k} = g^{f_i(ID_j)} (mod \ p) = y_{i,j} \tag{5.1}$$

3. Participants are disqualified if they do not follow the protocol correctly. The remaining set of participants form the set Q. The protocol aborts if Q contains less than t members. P_j uses the correct shares received from $i \in Q$ to compute the following:

$$x_j = \sum_{i \in Q} (s_{i,j} L_i) \ mod \ q, \tag{5.2}$$

where the Lagrangian coefficient is given as:

$$L_i = \prod_{k \in Q, k \neq i} \frac{ID_k}{ID_k - ID_i} \ mod \ q \tag{5.3}$$

The secret key x_j is P_j's *new* share in the distributed group secret key x_Q. The group public key can be computed (verified) by any group member or outsider as:

$$y_Q = \prod_{i \in Q} A_{i,0}^{L_i} = g^{\sum_{i \in Q} f_i(0) L_i} = g^{x_Q} \ mod \ p \tag{5.4}$$

Each P_j, $j \in Q$ calculates its public key as $y_j = g^{x_j} \ mod \ p$. Using the ElGamal type signature variant GES, P_j binds itself to the public key y_j by generating a signature (r_{P_j}, s_{P_j}) on y_j using its long-term private key SK_j. P_j broadcasts (y_j, r_{P_j}, s_{P_j}) to all protocol participants. P_j also verifies the authenticity of the received public keys after calculating the public key y_i of all P_i, $i \in Q$ as:

$$y_i = g^{x_i} = g^{\sum_{j \in Q} (s_{j,i} L_j)} = \prod_{j \in Q} y_{j,i}^{L_j} \ mod \ p \tag{5.5}$$

4. Each P_j, $j \in Q$ completely erases *all* traces of its initial randomly chosen value d_j and all generated subshares $s_{j,i} = f_j(ID_i)\ mod\ q$ $(i = 1, \ldots, n)$ and received subshares $s_{i,j}$ $(i \in Q)$.

The secret can be constructed if t or more honest players, from the set Q, collect:

$$x_Q = \sum_{j \in Q} (x_j \prod_{k \in Q, k \neq j} \frac{ID_k}{ID_k - ID_j})\ mod\ q \tag{5.6}$$

Taking the Lagrange polynomial of the original random values d_j, for $j \in Q$, also yields the group secret x_Q which highlights the importance of erasing these values securely.

In the following Section 5.3 a publicly verifiable *distributed-key redistribution/update* (DKRU) protocol is proposed that is compatible with the initial DKG protocol given above. The differences and the significance of the similarity are discussed in Section 5.4.2 and Section 5.5.

5.3 Proposed Publicly Verifiable Distributed-Key Redistribution/Update Protocol

The proposed publicly verifiable *distributed-key redistribution/update* (DKRU) protocol is as follows:

A distributed group secret x_Q is redistributed from $\Gamma_P^{n,t}$ to a *new* access structure $\Gamma_{P'}^{n',t'}$, using the shares of the authorized subset $\beta \in \Gamma_P^{n,t}$. The system parameter setup given in Section 5.2.1 is applicable.

The initial secret distribution to the access structure $\Gamma_P^{n,t}$ is performed using the publicly verifiable *distributed-key generation* (DKG) scheme given in Section 5.2.2. Note that the proposed DKRU protocol is aligned with the initial DKG protocol. The minor differences and the significance of the similarity are discussed in Section 5.4.2 and Section 5.5. The proposed DKRU protocol can be used as a publicly verifiable *distributed-key updating* (DKU) protocol by simply keeping the access structure constant, $\Gamma_{P'}^{n',t'} = \Gamma_P^{n,t}$. For the sake of proactive security the threshold cryptosystem shares should be periodically updated within a limited time period T.

The *publicly verifiable property* by definition means that any outsider can obtain all the required information from the broadcast channel to validate the operation of both the *initial* DKG and DKRU protocols.

Assume that a subset α of new members join or existing members leave the threshold cryptosystem.

The protocol executes in four steps:

1. P_i, $i \in \beta$ share their secret share x_i of group secret key x_Q with $P_j \in \Gamma_{P'}^{n',t'}$, using Shamir's secret sharing scheme [135]:

   For $i \in \beta$:

   (a) P_i sets $a'_{i,0} = s'_{i,0} = x_i$ and chooses at random $a'_{i,k} \in [1, q-1]$, for $1 \leq k \leq t'-1$, such that the numbers $a'_{i,0}, \ldots, a'_{i,t'-1}$ define a polynomial $f'_i(X) = \sum_{k=0}^{t'-1} a'_{i,k} X^k$ over $\mathbb{Z}_q$ of degree $t'-1$.

   (b) For $j = 0, \ldots, t'-1$, P_i computes, $A'_{i,j} = g^{a'_{i,j}} \ mod\ p$. For $j = 1, \ldots, n'$, P_i computes $s'_{i,j} = f'_i(ID_j) \ mod\ q$, $y'_{i,j} = g^{s'_{i,j}} \ mod\ p$ and an encryption $E_{PK_j}(s'_{i,j})$ of each secret shadow. P_i then binds itself to the public values by generating a digital signature (r'_{P_i}, s'_{P_i}) on message $([y'_{i,1}, E_{PK_1}(s'_{i,1})] \parallel \cdots \parallel [y'_{i,n'}, E_{PK_{n'}}(s'_{i,n'})] \parallel A'_{i,0} \parallel \cdots \parallel A'_{i,(t'-1)} \parallel y_\beta)$ using the ElGamal type signature variant: $GES = (M.EG\text{II}.3.\sigma(1), r, s, h(m,r), 1)$, developed from the generalized ElGamal scheme presented in [53]. The signature is generated with P_i's long-term private key SK_i. P_i broadcasts the message and signature to all protocol participants. Note that P_j, $j \in \beta$ knows the authentic public values y_i of $P_i \in \Gamma_P^{n,t}$, from a previous DKG or DKRU operation (see Eq.(5.5) and Eq.(5.10)). The set y_β is formed by y_i, $i \in \beta$ and only included in the broadcast message if *new* members join the group. The encryption $E_{PK_j}(s'_{i,j})$ of the secret shadow for protocol participant P_j is performed using an appropriate *publicly verifiable encryption scheme* [127] [137]. P_i keeps the share when $j = i$.

2. Protocol participants $P_1, \ldots, P_{n'}$ each verify the following:

   (a) (r'_{P_i}, s'_{P_i}) is a valid signature on the public values $([y'_{i,1}, E_{PK_1}(s'_{i,1}] \parallel \cdots \parallel [y'_{i,n'}, E_{PK_{n'}}(s'_{i,n'})] \parallel A'_{i,0} \parallel \cdots \parallel A'_{i,(t'-1)} \parallel y_\beta)$.

   (b)

   $$A'_{i,0} = y_i \ \forall\ i \in \beta \tag{5.7}$$

   If P_j, $j \in \alpha$ are *new* members joining the group then they must only use y_i's found to be consistant in the broadcast messages received from t or more members of subset β. If the conditions do not hold for more than t members from the authorized subset β,

abort the protocol, otherwise evaluate the following equation:

$$\prod_{k=0}^{t'-1} A_{i,k}'^{(ID_j)^k} = \prod_{k=0}^{t'-1} (g^{a'_{i,k}})^{(ID_j)^k} = g^{\sum_{k=0}^{t'-1}(a'_{i,k})(ID_j)^k} = g^{f'_i(ID_j)} (\ mod\ p) = y'_{i,j} \quad (5.8)$$

P_j knows that the *subshare* distribution received from P_i is correct if Eq.(5.8) holds.

3. Participants are disqualified if they do not follow the protocol correctly. The remaining set of participants form the set Q'. The protocol aborts if Q' contains less than t members of the authorized subset β. P_j uses the correct shares received from $i \in Q'$ to compute the following: $x'_j = \sum_{i \in Q'} s'_{i,j} L'_i$, where the Lagrangian coefficient is given as:

$$L'_i = \prod_{k \in Q', k \neq i} \frac{ID_k}{ID_k - ID_i}\ mod\ q \quad (5.9)$$

The secret key x'_j is P_j's *new* share in the distributed group secret key $x_{Q'}$[1]. If the conditions presented in Step-2 hold then P'_j knows that its *new* secret key x'_j is a valid share of x'_Q.

Each P_j, $j \in Q'$ calculates its public key as $y'_j = g^{x'_j}\ mod\ p$. Using the ElGamal type signature variant GES and its long-term private key SK_j, P_j generates a signature (r'_{P_j}, s'_{P_j}) on y'_j binding itself to the public value. P_j broadcasts $(y'_j, r'_{P_j}, s'_{P_j})$ to all network participants. P_j also calculates the public key of all P_i, $i \in Q'$ as:

$$y'_i = g^{x'_i} = g^{\sum_{j \in Q'}(s'_{j,i} L_j)} = \prod_{j \in Q'} y'^{L_j}_{j,i}\ mod\ p \quad (5.10)$$

4. Each P_j, $j \in Q'$ completely erases *all* traces of its old share x_j and all generated subshares $s'_{j,i} = f'_j(ID_i)\ mod\ q$ $(i = 1, \ldots, n')$ and received subshares $s'_{i,j}$ $(i \in Q')$.

The secret can be constructed if t' or more honest players, from the set Q', collect:

$$x_{Q'} = \sum_{j \in Q'} (x'_j \prod_{k \in Q', k \neq j} \frac{ID_k}{ID_k - ID_j})\ mod\ q \quad (5.11)$$

Any network participant can calculate (verify) the group public key when needed, using the available public information, as follows:

$$y_{Q'} = \prod_{i \in Q'} A'^{L'_i}_{i,0} = g^{\sum_{i \in Q'} f'_i(0) L'_i} = g^{x_{Q'}}\ mod\ p \quad (5.12)$$

[1]The group secret remains constant irrespective of distributed-key redistribution or distributed-key updates, $\therefore y_{Q'} = g^{x_{Q'}} = y_Q = g^{x_Q}$. Group members P_j, for $j \in Q'$ belonging to the new access structure $\Gamma^{n',t'}_{P'}$ thus share the same group secret and public key as the old access structure $\Gamma^{n,t}_{P}$.

5.4 Discussion on the Security and Features of the Proposed Distributed-key Management Infrastructure

The security analysis considers a straightforward *adversary model* as defined in [131]. An adversary is a malicious party that uses realistic means available to break the proposed scheme. Examples of such adversarial behavior are given in [131]. Any *active* adversary can eavesdrop on all the communication between group members, modify the content of messages and inject them back into the channel. When an honest party is compromised all its public and private information is exposed to the adversary.

The security of the *distributed-key generation* (DKG) scheme given in Section 5.2.2 is discussed next followed by the *distributed-key redistribution/update* (DGRU) scheme presented in Section 5.3.

It is generally accepted that the security of distributed-key distribution, update and redistribution protocols is defined in terms of *correctness* and *confidentiality* or *secrecy* [127] [130] [131].

5.4.1 Initial key sharing security

The security proof of the proposed round optimal DKG protocol is similar to the scheme of Zhang *et al.* [131] and discussed here for the sake of completeness.

To argue the security of the proposed DKG scheme we first claim the correctness of the scheme against adversaries. This means that the following should hold true in the context of the proposed scheme [130] [129] [131]:

Correctness Property 1 All subsets of at least t shares provided by honest parties (from set Q) define the same, unique distributed-key (secret) x.

Correctness Property 2 All honest parties have the same public key $y_Q = g^{x_Q}$. This should not be disturbed by a malicious adversary, hence guaranteed by the previous item.

Correctness Property 3 The secret key is uniformly distributed in the key space. That is, each element of the group is chosen as the secret key with equivalent probability.

Correctness Property 1 is clear to see from Equation 5.6 and Property 2 follows from Property 1. The selection of x_Q as $a_{i,0} = s_{i,0} = d_i \in [1, q-1]$ and sharing the value x_Q as per Shamir's secret

sharing scheme validates Correctness Property 3.

Next we claim that the DKG protocol protect the secrecy of x_Q. This means than an adversary that learn less than t shares of the secret key (by corrupting less than t honest parties) gain no information on x_Q beyond what follows from $y_Q = g^{x_Q}$ [130]. As per [130] [131] it is easy to build a simulator which run a pre-decided program, in expected polynomial time, in stead of a real protocol run [131]. Such a simulator is defined in [130] and will not be repeated here. Given Correctness Property 3, it is trivial to see, from the view of an adversary, that the output of the simulator is computationally indistinguishable from a real protocol run. We therefore conclude that the DKG scheme presented in Section 5.2.2 satisfies both the *correctness* and *confidentiality* security requirements.

5.4.2 Proactive security and secret redistribution

The proposed DKMI defends against mobile/active adversaries by proactively updating the group key shares every period T. To allow for *dynamic* group membership and modification of the availability/security tradeoff, the shares can be redistributed to a new access structure $\Gamma_{P'}^{n',t'}$. The proposed publicly verifiable *distributed-key redistribution/update* (DGRU) protocol proposed in Section 5.3 was designed to support dynamic group membership and allow for share updates by merely keeping the access structure constant, i.e., $\Gamma_{P'}^{n',t'} = \Gamma_{P}^{n,t}$. Comparing the DGRU protocol to the *initial* DKG protocol presented in Section 5.2.2, the reader will note that these protocols are almost exactly equivalent. By deliberately keeping the phrasing and notation as far as possible the same, the following minor differences are easily identifiable:

- For the *initial* DKG each user must choose a random number d_i, while the DKRU protocol uses the user's share x_i of the group secret x_Q.
- Members of the authorized subset β append the public values y_i of $P_i \in \beta$ to $([y'_{i,1}, E_{PK_1}(s'_{i,1})] \parallel \cdots \parallel [y'_{i,n'}, E_{PK_{n'}}(s'_{i,n'})] \parallel A'_{i,0} \parallel \cdots \parallel A'_{i,(t'-1)})$ if *new* members join the group.
- In the DKRU protocol the participants must also check that the condition presented in Eq.(5.7) holds, before verifying whether the *subshare* distribution is correct by using Eq.(5.8). New members should only use public values that have been found to be consistent in t or more broadcast messages. If *both* Eq.(5.7) and Eq.(5.8) are satisfied for t or more members of β, the users are guaranteed that their *new* share in the *original* group key x_Q is valid.

The DKG protocol due to Zhang *et al.* [131] was modified to complement the proposed DKRU protocol given in Section 5.3 and to allow for practical key redistribution/updating. Besides the differences given above, the *initial* DKG protocol, DKU protocol and DGR protocol are conveniently performed by a single, publicly verifiable, round optimal protocol. A protocol suite capable of servicing *all* aspects of secret sharing is defined here as a ***distributed-key*** *management infrastructure* (DKMI).

The proposed DKMI presented in Section 5.2.2 and Section 5.3 solves a major problem with existing secret *redistribution/update* protocols. This problem is identified by Wong *et al.* [134]: by definition a threshold cryptosystem allows t or more members holding a secret share to reconstruct the group secret. In a proactively secure threshold cryptosystem not more than $t-1$ members can be compromised or become faulty within an update time period T, otherwise the system is broken. Referring to the proposed DKRU protocol presented in Section 5.3 and the secret redistribution scheme by Wong *et al.* [134], the authorized subset β may thus contain, in the worst case scenario, $t-1$ malicious members out of the total n group members. The $t-1$ malicious members in β can thus distribute subshares $\overline{s_{i,j}}$ of an incorrect share $\overline{x_i}$ to new members joining the system. Although new members can detect the discrepancy in the broadcast values, they cannot identify which members are malicious. It is noted here that this is not only applicable to the new members in the scheme proposed by Wong *et al.*, but also *existing* members cannot identify the malicious members in β and therefore malicious members cannot be disqualified. Consequently the redistribution protocol proposed by Wong *et al.* [134] must be repeated each time with a different authorized subset β', with $\mid \beta' \mid = t$, until all values are consistent and all the verification conditions hold. Wong *et al.* give a worst case repetition of the protocol bounded by $\binom{n}{t} - \binom{n-t+1}{t} = \sum_{i=1}^{t-1} \binom{t-1}{i}\binom{n-t+1}{t-i}$, which makes the scheme impractical.

This chapter presents a solution to the above problem with existing secret *redistribution/update* protocols as follows:

In the proposed DKMI the protocol participant P_j in the *initial* DKG protocol uses Eq.(5.5) to calculate the authentic public values y_i corresponding to the secret share x_i of each P_i, for $i \in Q$. In the DKRU protocol, members of the authorized subset β construct a polynomial with their secret share set to $a'_{i,0} = s'_{i,0} = x_i$. In order to ensure that Eq.(5.8) holds and to avoid identification, the malicious members are forced to compute and broadcast $A'_{i,0} = g^{a'_{i,0}}$, with their computed subshare witnesses $y_{i,j}$. Assume that a subset α of new members are joining the threshold cryptosystem. P_i, $i \in \beta$ broadcast to all members a message that includes their calculated public keys y_i, for all other P_i, $i \in \beta$. All existing members forming part of the new access structure $\Gamma_{P'}^{(n',t')}$ can

detect and positively identify a malicious member that distributes subshares $\overline{s_{i,j}}$ of an incorrect share $\overline{x_i}$, by checking if Eq.(5.7) holds. The only feasible option is to require the authorized subset β to include all the existing members that will form part of the new access structure $\Gamma_{P'}^{n',t'}$. By the fundamental definition of a threshold cryptosystem this will always ensure at least t members of β with consistent public values. This requirement places no limitation on the practicality of the proposed distributed-key management infrastructure (DKMI)[2] and keeps the scheme round optimal.

The new members will detect the discrepancies in the public values broadcast by the members in β. Since the new members will receive consistent public values from at least t participants, the participants with inconsistent values can therefore be considered malicious or faulty and are therefore positively identified[3]. Before joining members have obtained a *valid* share of the group secret they do not have any authority and therefore can only *implicitly* aid in the disqualification of malicious members by only using the subshares of the t or more members with consistent public values to compute their own *new* share in the group key.

Finally, providing proof of the *correctness* and *confidentiality* security requirements for the proposed DKRU protocol is similar to that of the DKG protocol given in Section 5.4.1.

5.4.3 Efficiency analysis

The *worst case* computational cost of the *initial* DKG protocol and DKRU protocol are briefly considered.

Efficiency analysis on the initial DKG protocol (Section 5.2.2) and DKRU protocol (Section 5.3) shows that these protocols require $O(nt)$ and $O(n't')$ exponentiations respectively. Multiplications yield a similar growth rate to the exponentiations. Each protocol participant is required to generate random numbers $O(t)$ for initial DKG and $O(t')$ for DKRU.

The threshold cryptosystem accommodates a total of n^2 shares. Each participant broadcasts $O(t) + O(n)$ messages for initial DKG and $O(t') + O(n')$ for DKRU. The network as a whole has a

[2]It is obvious that existing members that do not participate in the distributed-key redistribution *or* distributed-key update procedure will be excluded from the threshold cryptosystem, therefore old members wanting to form part of P' are always available to help with distributing subshares to new members.

[3]The proposed scheme does have another mechanism for *all* network participants to identify malicious members. Since both the *initial* DKG protocol (Section 5.2.2) and DKRU protocol (Section 5.3) are both *publicly verifiable*, all network participants have exactly the same information as existing members of the threshold cryptosystem, except for a valid share in the group secret. A joining member who has observed a periodic DKRU procedure can calculate the authentic public values of each group member using Eq.(5.10) and therefore identify malicious members during the next DKRU procedure.

communication cost of $O(nt)+O(n^2)$ and $O(n't')+O(n'^2)$ for initial DKG and DKRU respectively.

5.5 Conclusion

The chapter address the issue of *initial* distributed-key generation (DKG) to construct a threshold cryptosystem without the assistance of a trusted authority. To realize initial DKG, improvements were made to the round optimal DKG scheme of Zhang *et al.* The chapter proposes a novel publicly verifiable, round optimal (one round) distributed-key redistribution/updating (DKRU) protocol that eliminates a major problem with existing schemes; malicious or faulty protocol participants from the original access structure are positively identifiable by the existing honest members as well as the new members joining the threshold cryptosystem, which avoids repeating the secret redistribution protocol until all the verification conditions holds.

The chapter proposes the first integrated (single protocol) solution for DKG, distributed-key updating (DKU) and distributed-key redistribution (DKR). Although the initial DKG protocol and DKRU protocol are presented separately for clarity it was shown that the two protocols are essentially equivalent. The same *initial* DKG protocol can thus be used to realize DKR and DKU by merely keeping the access structure constant. The chapter defines the notion of a *distributed-key management infrastructure* (DKMI) as a protocol suite that considers all aspects of secret distribution (DKG), secret updating (DKU) and secret redistribution (DKR). Considering the minor differences between the *initial* DKG protocol and DKRU protocol, the proposed DKMI requires considerable less implementation effort than implementing three unrelated protocols to realize distributed-key management.

Use of the DKRU mechanism makes the proposed *fully* distributed threshold-multisignature scheme proactively secure, allows for *dynamic* group membership and gives the group members the capability of adjusting the availability/security tradeoff by redistributing the existing access structure $\Gamma_P^{(n,t)}$ to a new access structure $\Gamma_{P'}^{(n',t')}$.

Part IV

Threshold-Multisignatures for Distributed Communication Systems

Chapter 6

A Fully Distributed Proactively Secure Threshold-Multisignature Scheme

6.1 Introduction

Threshold-multisignature schemes [15] combine the properties of threshold *group-oriented* signature schemes [125] and *multisignature* schemes [138]. In literature, threshold-multi-signature schemes are also referred to as threshold signature schemes with *traceability* [139] [140] [141]. The combined properties guarantee the signature verifier that at least t members participated in the generation of the group-oriented signature and that the identities of the signers can be easily established. The majority of existing threshold-multisignature schemes belong to variants of the single signatory, *generalized ElGamal signatures* [53] [142], extended to a group/multiparty setting.

In [143], Wang defines the properties of threshold *group* signature schemes [144] in order to guarantee the security of the system. To the best of the author's knowledge a complete set of definitions does not exist for *threshold-multisignature* schemes. The author's studies have shown that secure threshold-multisignature schemes must satisfy the following five main properties:

Correctness: All threshold-multisignatures on an arbitrary message m, generated by an honest authorized subset β of group members, forming subgroup P_β, can be verified by any outsider V

(with respect to the group). This implies that the group-oriented signature is publicly verifiable.

Threshold property: Only a threshold of t or more *authorized* group members are able to collaboratively generate a valid threshold-multisignature. This property thus incorporates *unforgeability*.

Traceability: Any outsider V can learn the identities of the individual signers belonging to P_β from the threshold-multisignature on m without interaction with any of the group members and/or a group manager. This implies that the signers are publicly traceable with public information. Traceability implies accountability; the individual signers participating in the threshold-multisignature scheme can be held accountable for their contribution to the group-oriented signature.

Coalition-resistance: No colluding subset of group members can generate a valid threshold-multisignature not satisfying the traceability property. Coalition-resistance subsumes *framing-resistance*, i.e. no subset of group members can sign on behalf of any other subset of group members.

Break-resistance: An adversary in possession or control of the group secret key and/or the individual secret shares of any number of group members, cannot generate a valid threshold-multisignature and/or partial/individual signatures, thus although the underlying *threshold cryptosystem* has been broken, the threshold-multisignature signature scheme should not be breakable.

Threshold-multisignature schemes can be differentiated from threshold *group* signatures [144] by the fact that by definition in the latter the individual signers remain anonymous, since it is computationally hard to derive the identities from the group signature with the exception of the group managers. In contrast, by the above defined *traceability property* of threshold-multisignature schemes, the individual signers are publicly traceable and do not enjoy anonymity. Consequently the traceability property of threshold-multisignature schemes allows the individual signers to be held accountable in the public domain and renders the *unlinkability* property of threshold *group* signature schemes as defined in [143], inapplicable.

The main objective of this chapter is to propose a new *threshold-multisignature* scheme without a trusted third party (TTP), based on the ***distributed-key*** *management infrastructure* (DKMI) presented in Chapter 5. The proposed scheme, originally presented in [18], can be easily adapted to incorporate a TTP; a version of the proposed scheme with the assistance of a TTP will therefore not be presented. The proposed discrete logarithm based threshold-multisignature scheme is also *proactively* secure allowing for DKR to a new access structure $\Gamma_{P'}^{(n',t')}$ and periodic DKU to mitigate attacks from an active/mobile adversary. The proposed discrete logarithm based threshold-

multisignature scheme is made proactively secure, as explained in Chapter 5, by periodically updating secret shares and facilitating changes in group membership by allowing an authorized subset β of existing group members to redistribute secret shares to a new access structure $\Gamma_{P'}^{(n',t')}$.

The chapter is organized as follows: In Section 6.2 the new threshold-multisignature scheme is proposed. The security and features of the proposed threshold-multisignature scheme are discussed in Section 6.4. Some conclusions are provided in Section 6.5.

6.2 Proposed Threshold-Multisignature Scheme

In this section a novel threshold-multisignature scheme, without assistance from a trusted key distribution center (KDC), is proposed. The scheme consists of the following six parts (A-F):

6.2.1 System model

We assume the same system model defined in Chapter 5.

6.2.2 Initial publicly verifiable distributed-key generation

We use the *distributed-key generation* (DKG) protocol is presented in Chapter 5 to realize secure, *initial* secret sharing to an access structure $\Gamma_P^{n,t}$, without a trusted dealer or KDC.

6.2.3 Individual signature generation

Any subset β of t or more members can represent the group and sign an arbitrary message m. Each member P_i, $i \in \beta$ selects a random integer $k_i \in [1, q-1]$ and computes $r_i = g^{k_i} \ mod \ p$. Each member *verifiably encrypts* k_i with its *own* long-term public key PK_i using a *verifiable encryption scheme* [127] [137] to generate $E_{PK_i}(k_i)$. Using the ElGamal type signature variant GES, P_i generates a signature (r'_{P_i}, s'_{P_i}) on $[r_i, E_{PK_i}(k_i)]$ using its long-term private key SK_i. P_i broadcasts $(r_i, E_{PK_i}(k_i), r'_{P_i}, s'_{P_i})$ to all protocol participants. This implies that each member commits to its public value r_i and provides a proof of knowledge of its corresponding discrete logarithm, k_i.

After all committed r_i's are available and members with invalid r_i's are excluded from the set β, the value R is calculated as follows:

$$R = \prod_{i \in \beta} r_i \ mod \ p \tag{6.1}$$

Each P_i uses public values ID_j authentically bound to PK_j, for $j \in \beta$ to form the set B.

P_i uses its (secret share)/(private key) set (x_i, SK_i) and random number k_i to compute its individual signature s_i on message m as follows:

$$s_i = [H(m, R, B)][(x_i) \cdot L_{\beta i} + SK_i] + k_i \ mod \ q, \tag{6.2}$$

where the Lagrange interpolating coefficient $L_{\beta i}$ is given by:

$$L_{\beta i} = \prod_{k \in \beta, k \neq i} \frac{ID_k}{ID_k - ID_i} \ mod \ q$$

The set $(s_i, r_i, L_{\beta i}, B)$ is the individual signature of P_i on message m, which is broadcast to all other group members.

6.2.4 Individual signature verification

On receiving all the signatures $(s_i, r_i, L_{\beta i}, B)$, P_j, $j \in \beta$, performs the functionality of a clerk and uses the public key set (y_i, PK_i) to authenticate the individual signature of P_i by verifying if $L_{\beta i}$ is correct and whether the following equation holds for $i \in \beta$, $i \neq j$:

$$g^{s_i} = (y_i^{L_{\beta i}} PK_i)^{[H(m,R,B)]} \ r_i \ mod \ p \tag{6.3}$$

If Eq.(6.3) fails to hold, the individual signature of P_i on message m is invalid. Participants are disqualified if their individual signatures are found to be invalid. The remaining honest participants form the set α and repeat the *individual signature generation* part from Eq.(6.1), with $\beta = \alpha$. The protocol aborts if α contains less than t members.

6.2.5 Threshold-multisignature generation

After P_j, $j \in \alpha$ has received and verified t or more individual signatures the second signature parameter S of the threshold-multisignature (R, S) on message m can be computed as:

$$S = \sum_{i \in \alpha} s_i \ mod \ q \tag{6.4}$$

The set of identities B is appended to (R, S) and provides an explicit link between the threshold signature and the subset of individual signers who collaborated to generate the threshold signature (R, S, B) on message m.

6.2.6 Threshold-multisignature verification and individual signer identification

Any outsider can use the group public key y_Q and public keys PK_i bounded to the identities ID_i, for $i \in \alpha$ to verify the validity of the threshold-multisignature (R, S, B) on an arbitrary message m. The signature verifier calculates the subgroup public key PK_α as follows:

$$PK_\alpha = g^{\sum_{i \in \alpha} SK_i} = \prod_{i \in \alpha} PK_i \ mod \ p \tag{6.5}$$

The signature verifier now has all the information to check if the following congruency holds:

$$g^S \equiv (y_Q PK_\alpha)^{[H(m,R,B)]} R \ mod \ p \tag{6.6}$$

If Eq.(6.6) holds, the threshold-multisignature (R, S, B) on message m is valid and the subgroup B of individual signers are positively identified. (See Section 6.4.1 for *proof of correctness*.)

6.3 Proposed Publicly Verifiable Distributed-Key Redistribution/Update Protocol

The proposed publicly verifiable *distributed-key redistribution/update* (DKRU) protocol defined in Chapter 5 is used to update or redistribute the group secret x_Q from $\Gamma_P^{n,t}$ to a *new* access structure $\Gamma_{P'}^{n',t'}$, using the shares of the authorized subset $\beta \in \Gamma_P^{n,t}$.

6.4 Discussion on the Security and Features of the Proposed Threshold-Multisignature Scheme

The proposed threshold-multisignature scheme is based on a multiparty extension of the ElGamal type signature variant: $GES = (M.EG\text{II}.3.\sigma(1), r, s, h(m, r), 1)$ [53]. The proposed threshold-multisignature scheme can equally use any other secure and efficient signature variant of the ElGamal type signature scheme. References [53] [142] help in selecting such a variant. The main reason for using the defined GES is to minimize the computational cost of generating and verifying the individual signatures and group-oriented signature in a multiparty setting, without compromising security.

The security analysis considers a straightforward general *adversary model*. An adversary is a malicious party that uses every means available to break the proposed threshold-multisignature scheme. Any *active* adversary can eavesdrop on all the communication between group members, modify the content of messages and inject them back into the channel. When an honest party is compromised all its public and private information is exposed to the adversary.

To argue the security and validity of the proposed threshold-multisignature scheme, it is enough to show that the scheme fulfills all the fundamental properties of generic threshold-multisignature schemes given in Section 6.1 and resists attacks to which other similar schemes are vulnerable.

6.4.1 Correctness and threshold property

Any t or more honest group members of the authorized subset α can collaborate and use the threshold-multisignature scheme to produce a valid threshold-multisignature (R, S, B) on an arbitrary message m. In the context of the statement, *honest* implies that the group members follow the protocol as specified in Section 6.2.

Proof. $s_i = [H(m, R, B)][(x_i) \cdot L_{\alpha i} + SK_i] + k_i \ mod\ q \ (i \in \alpha)$
$S = \sum_{i\in\alpha}(s_i) = [H(m, R, B)] \sum_{i\in\alpha} [(x_i) \cdot L_{\alpha i}] + [H(m, R, B)] \sum_{i\in\alpha} (SK_i) + \sum_{i\in\alpha} (k_i) \ mod\ q$
$g^S = g^{[H(m,R,B)][\sum_{i\in\alpha}[(x_i)\cdot L_{\alpha i}]]} g^{[H(m,R,B)]\sum_{i\in\alpha}(SK_i)} g^{\sum_{i\in\alpha}(k_i)} \ mod\ p$
$= g^{[H(m,R,B)]x_Q[\prod_{i\in\alpha}(PK_i)]^{[H(m,R,B)]} \prod_{i\in\alpha}(r_i)} \ mod\ p$
$= (y_Q PK_\alpha)^{[H(m,R,B)]} R \ mod\ p$ □

6.4.2 Traceability of signers

The reason for the intractability of [139] and [141], as presented by [145] and [146] [147] respectively, is due to the nonexistence of a relationship between the threshold signature (R, S) and the Lagrange polynomial $h(y)$ used by the signature verifier to identify the individual signers. The polynomial $h(y)$ is constructed by the designated combiner with t pairs of public values belonging to members of the subgroup β that submitted valid partial signatures. The Lagrange interpolation polynomial, $h(y)$ is given as [139]:

$$h(y) = \sum_{i=1}^{t} x_i \prod_{j=1, j\neq i}^{t} \frac{y - y_j}{y_i - y_j} = b_{t-1}y^{t-1} + \ldots + b_1 y + b_0 \tag{6.7}$$

It is noted here that it is possible to create such a relationship between $h(y)$ and (R, S) and make the schemes proposed in [139] [141] traceable, by including $h(y)$ within the *individual signatures*. It is in the view of the author that the schemes presented in [139] [141], with a strong binding between $h(y)$ and (R, S), still fail to provide an optimum solution to ensure traceability. Using the polynomial function $h(y)$ to capture the signers' identities results in additional computational overhead for the threshold signature verifiers ($n(t-1)$ multiplications and $n(t-2)$ exponentiations) and fails to reduce the threshold group-oriented signature size. The polynomial $h(y)$ of degree $t-1$, constructed from t data points, requires at least t coefficients $(b_0, b_1, \ldots, b_{t-1})$ to be uniquely defined. Rather than including the coefficients in the individual signatures and appending the coefficients to the threshold signature, the proposed threshold-multisignature scheme uses the subset of identities B instead. This saves the signature verifier the computational overhead of evaluating $h(y)$ to learn the identities of the individual signers. The computational cost of generating $h(y)$ is considered in Section 6.4.7.4.

The traceability property of the proposed threshold-multisignature scheme is verified in Section 6.4.3 by showing that the signature verifier can only validate the threshold signature if an authentic subset B of individual signers collaborated to generate the signature. Since the subset B of identities is used to validate the threshold signature (Eq.(6.6)) the individual signers are publicly traceable and explicitly bound to the threshold signature.

If a verifier were to know and use the certified public keys of the individual signers, why not simply use a list of individual (RSA) signatures for the group signature?

The threshold property must be guaranteed by cryptographic means during the generation of the threshold group-oriented signature to effectively share the power of the cryptosystem between the

group members [124] [125]. The actual value of the threshold t may be (dynamically) determined by an authorized subset of group members [133] and thus forms part of the group policy. Expecting that all (future) threshold group-oriented signature verifiers will consistently enforce a (dynamic) group policy makes the system vulnerable to attack.

6.4.3 Coalition-resistance and break-resistance

The coalition- and break-resistance properties can be verified by showing that the proposed threshold-multisignature scheme is break-resistant:

Assume a malicious subgroup β of t or more members conspire to obtain the group secret x_Q and wants to frame valid subgroup α for a signature on an arbitrary message m. This implies β wants to generate a valid untraceable group-oriented signature. Any malicious party P_j chooses a random number $k \in [1, q-1]$ and generates $R = g^k$. P_j uses the publicly known values (ID_i), for $i \in \alpha'$ to form the subgroup A. (The public value PK_α can easily be calculated using Eq.(6.5).) The malicious member can attempt to forge the threshold-multisignature by computing $S = [H(m, R, A)](x_Q + SK_\alpha) + k$, which is impractical since P_j cannot compute SK_α from PK_α based on the intractability of the discrete logarithm problem. An alternative method is to solve for S to satisfy the verification equation $g^S = (y_Q PK_\alpha)^{[H(m,R,A)]} R \ mod \ p$ which is again impractical based on the intractability of the discrete logarithm problem. An attacker's last resort is to randomly select a value F and signature parameter S and then attempt to determine a value R' that satisfies $g^S = (y_Q PK_\alpha)^F R \ mod \ p$ and $F = [H(m, R', A)]$ simultaneously. This is known to be impractical based on the properties of a secure collision free one-way hash function $H(\cdot)$.

This shows that using the proposed threshold-multisignature scheme to produce a valid untraceable signature or signing on behalf of any other subset of group members is not possible even if the threshold cryptosystem is broken, i.e. the group secret x_Q is known or controlled by an adversary. It also confirms that the individual signers that collaborate to generate a threshold-multisignature are always traceable. With little modification to the above argument it can be shown that forging *individual signatures* is also impractical, even if a coalition of t or more group members conspire to obtain an individual's partial share x_i of the group key x_Q[1]. It can thus be concluded that the proposed threshold-multisignature is *break-resistant*.

[1]A formal security proof for the defined ElGamal signature variant *GES* in the Random Oracle and Generic Model (ROM+GM) can be found in [51]. The security proof of the underlying individual signature scheme supports the break-resistance property.

6.4.4 Attacks on threshold-multisignature schemes

6.4.4.1 Collusion attack

A *collusion attack* enables dishonest group members (and/or a designated clerk or combiner node: see Section 6.4.5), that work together, to control the group key.

Wang *et al.* [147] report a collusion attack on the threshold-multisignature scheme (without a trusted third party) presented by Li *et al.* in [141]. Wang *et al.* [147] propose some countermeasures against the attack. The first solution requires each member to publish its public value y_i simultaneously to mitigate the *rushing attack* on the group secret. This solution is impractical since there is no method, in existing networking protocols, of accommodating n simultaneous broadcasts nor can devices receive n simultaneous messages without advanced hardware. Any network participant will therefore always be able to make a legitimate excuse for a late arriving public value. The second proposed solution requires members to commit to their public values and then open their commitments to reveal the public values. This solution however makes the protocol interactive and thus vulnerable to an *adaptive* adversary [130] [131]. The third solution requires members to provide a non-interactive proof of knowledge of the discrete logarithm for the public value y_i to the base of the primitive element g. This solution is well known, and has been used by both Fouque *et al.* [130] and Zhang *et al.* [131] to eliminate the need for a *complaint phase*. As pointed out by Zhang *et al.* this approach was first introduced in [127] by Stadler to realize a *publicly verifiable* secret sharing scheme. The *complaint phase* also makes distributed-key generation (DKG) schemes such as [14] [126] [148] impractical considering the complexity of current multiparty computations [130] [131].

The proposed threshold multisignature scheme prevents adversaries from controlling the group key by using the publicly verifiable, one round DKG protocol, given in Section 6.2.2 and thus inherently eliminates attacks from an adaptive adversary during the construction of the threshold cryptosystem. The proposed scheme uses *publicly verifiable encryption* [127] [137] that allows protocol participants to provide a zero knowledge proof of their public values' discrete logarithm. Publicly verifiable encryption however does not offer a mechanism to bind participants to their public values. Committing participants to their public values and ensuring the public values' integrity are both essential if members are to be disqualified if their public values fail to satisfy the verification equations (for example Eq.(5.1) or Eq.(6.3)). The existing round optimal DKG protocols [130] [131] and threshold-multisignature schemes [15] [139] [140] [141] fail to guarantee a strong binding between participants and their public values. The weak binding allows an adversary

to manipulate public values that will result in the disqualification of honest participants. The proposed threshold-multisignature scheme and DKG protocol use the ElGamal type signature variant GES to generate secure and efficient digital signatures, which explicitly binds participants to their public values and simultaneously provides a mechanism to ensure the values' integrity.

6.4.4.2 Universal forgery attack

In [145], Tseng *et al.* present a universal forgery attack on the threshold signature schemes with traceability by Wang *et al.* [139]. The attack allows any group member or outsider to generate a valid forged threshold signature (R', S') on an arbitrary message m' from an existing valid threshold signature (R, S) on message m. Tseng *et al.* also show in [145] that Wang *et al.*'s schemes [139] are completely insecure since the trapdoor information $\alpha^{f(0)}$, which is in fact the group secret $(y = g^{\alpha^{f(0)}})$, can be calculated as $\alpha^{f(0)} = SR^{-1}H(m)^{-t}$ from any valid threshold signature (R, S) in message m.

The scheme by Wang *et al.* [139] is an example of an insecure ElGamal type signature variant extended to a multiparty setting. The universal forgery attack on [139] is a direct consequence of the insecurity of the *modified* ElGamal signature scheme on which the group signature is based. Caution should be taken in the modification of the ElGamal signature or when choosing an appropriate variant. The modification of the basic ElGamal signature scheme for the sake of efficiency can easily introduce an insecurity. The threshold-multisignature scheme proposed in Section 6.2 takes this fact into consideration. It is developed from a secure, and optimally efficient variant, that was selected using the guidelines given in [53] [142].

6.4.4.3 Rushing attack

In the context of group-oriented signature schemes a rushing attack is defined as a manipulation of the signature parameters or keying material by an adversary based on the public values received from all the other participants. The adversary thus waits until all other participants have played before defining its own value [130]. It is noted that the collusion attack by Wang *et al.* [147] and universal forgery attack by Wu *et al.* [146], both rely directly on a rushing attack. Assuming the threshold signature scheme is based on a secure and efficient variant [53] [142], it should be clear that the majority of attacks on group-oriented signature schemes can thus only come from a vulnerability introduced by the extension of the ElGamal type signature variant to accommodate multiple signers. Since group members must all make a valid contribution to the threshold

signature parameters and group secret, it is evident that manipulation of these values can occur if an adversary can delay playing its own contribution. The adversary can collect the contributed values of all other participants and then compute its own contribution to force the signature parameters or group secret to a known value when calculated as a function of all contributed values. Fouque *et al.* [130] and Zhang *et al.* [131] construct their distributed-key generation schemes on a *synchronous* network to prevent the adversary from delaying its response, thereby eliminating rushing attacks. Fouque *et al.* propose a solution to mitigate rushing attacks without using a *synchronous* network. Fouque *et al.* define an Incorruptible Third Party (ITP) to always play at the end, therefore eliminating the requirement for synchrony and preventing a malicious player from manipulating the signature parameters. The ITP unfortunately becomes a single point of vulnerability.

It is noted here that synchrony is not always required in a discrete logarithm based threshold group-oriented signature scheme based on a variant of the generalized ElGamal type signature. Forcing players to provide a zero knowledge proof of the discrete logarithm of all public values is sufficient to completely eliminate the rushing attack in an *asynchronous* network. Take for example the signature parameter R, which is generally calculated as $R = \prod_{i \in \beta} r_i \; mod \; p$, where $r_i = g^{k_i}$ is the public value of each player P_i with corresponding randomly chosen secret k_i. The malicious player P_j waits for each P_i to play its r_i. P_j chooses a random value k and sets $R = g^k$. After receiving all r_i's for $i \in \beta, i \neq j$, P_j factors out its public value as $r_j = R \prod_{i \in \beta, i \neq j} \; r_i^{-1} mod \; p$, which will force the other players to compute R at a later stage (of which the discrete logarithm is known by P_j). Note that P_j does not know the discrete logarithm of its public value r_j nor can it be calculated based on the intractability of discrete logarithms in finite fields. This will in particular be true if the modular arithmetic is done in an appropriate multiplicative subgroup $((Z)_p^*)$ or a cyclic subgroup of order q.

The attack presented by Michels *et al.* [149] on the threshold-multisignature proposed by Li *et al.* [15] is a good example of the rushing attack principle. As in the above example the malicious player P_j in this attack does not know the discrete logarithm of its own share r_j, thus, if the threshold group-oriented signature protocol requires each player to provide a zero knowledge proof of all public values' discrete logarithms, the benefit the adversary gains from a rushing attack, in an *asynchronous* network, is nullified. It will also prevent a malicious participant from disrupting the protocol, since the participant can simply be disqualified if it does not play a fair game.

The proposed threshold multisignature scheme in this paper requires players to provide a zero knowledge proof of the discrete logarithm of their public values by generating a *publicly verifiable*

encryption [127] [137] on the discrete logarithm of the public values. With this mechanism the proposed scheme eliminates rushing attacks without any requirement for synchrony.

6.4.4.4 Conspiracy attack

It is noted here that the threshold signature scheme proposed by Li *et al.* [141] is also vulnerable to a conspiracy attack by t or more malicious members. Any t or more members forming the subgroup β' can use their subshares $f(ID_i)_j$, $(i, j \in \beta')$ to construct the group secret key $f(0)$ as $f(0) = \sum_{j=1}^{t}(f(ID_i)_j \prod_{k=1,k\neq j}^{t} \frac{ID_j}{ID_j - ID_i})$. Any malicious member generates $R = g^{k_i}$ and solves the following congruence for integer S, $SR \equiv (R + h(m))k_i + f(0)\ mod\ q$. The set (R, S) will be a universally forged signature on the arbitrary message m, verifiable with the group-oriented signature verification equation, $g^{SR} = R^{(R+h(m))}y\ mod\ p$.

As shown in Section 6.4.3, conspiracy attacks in the proposed threshold-multisignature schemes are avoided by each member contributing an individual secret (long-term private key SK_i) to the group-oriented signature known to be *explicitly* associated with the member. A mechanism is provided for verifying the members' secret contributions from the threshold-multisignature, without revealing any additional information of the secrets, except the information that is publicly know to be associated with the secrets, i.e., the members' public keys PK_i, for $i \in \alpha$. Li *et al.* [15] defend against conspiracy attacks in a similar approach by adding a random number to the secret key, which effectively conceals the member's secret share of the group key. Note that this is however only accurate for Li *et al.*'s scheme *with* a trusted authority. Since the random numbers are not *explicitly* linked to the member's identity, the scheme presented in [15] does not have a strong *traceability property* [139].

Conspiracy attacks in threshold-multisignature schemes can also be avoided if the secret shares of members are generated in such a way that construction of the threshold cryptosystem's secret polynomial or derivation of the group secret from the members' secret shares, is computationally infeasible. Wang *et al.* [139] propose such a scheme, which prevents conspiracy attacks without attaching a random secret to secret shares. It is however noted that since a member's public value in [139] is not explicitly linked to a secret known to be associated with a group member, the scheme by Wang *et al.* also lacks a strong traceability property. The same comment can be made on the scheme proposed by Lee *et al.* in [140].

6.4.5 Symmetric relationships - eliminating the combiner

A centralized combiner node is a specialized server which can be seen as a single point of vulnerability and should be replicated in distributed networks (for example ad hoc networks) to ensure a correct combination [3]. The scheme proposed in this paper eliminates the need for a designated combiner/clerk as the construction of the threshold-multisignature is done by the shareholders themselves. This eliminates attacks relying on a corrupt clerk and mitigates the problem of ensuring the availability of a combiner node in distributed networks [3]. The scheme also preserves the symmetric relationship between the shareholders by placing on each node the same computational, memory and communication overhead.

6.4.6 Proactive security and secret redistribution

The proposed threshold-multisignature scheme defends against mobile/active adversaries by proactively updating the group key shares every period T. To allow for *dynamic* group membership and modification of the availability/security tradeoff, the shares can be redistributed to a new access structure $\Gamma_{P'}^{n',t'}$. The proposed publicly verifiable *distributed-key redistribution/update* (DGRU) protocol presented in Chapter 6.3 support dynamic group membership for the proposed threshold-multisignature scheme.

The security of the DGRU scheme is discussed in Chapter 5.4.2 and will not be discussed here any further.

6.4.7 Efficiency analysis

The efficiency of threshold-multisignatures may be based on the following six criteria:

6.4.7.1 Group public key length

The public key length in similar schemes will be briefly considered. In order to make a feasible comparison, only schemes that attempt to eliminate conspiracy attacks are evaluated:

Li *et al.* [15]: The group public key of [15] does not only consists of y, but also implicitly includes the following public values $\{x_i, z_{ij}\}$. In the Li *et al.* scheme, protocol participants attach a secret random number to their group secret shares in an attempt to avoid a *conspiracy attack*.

Consequently the public key includes the public values $\{x_i, z_{ij}\}$ since these values are required by the signature verifier to validate a threshold signature. It can thus be concluded that the public key is dependent on the number of group members n.

Wang *et al.* [139]: The scheme proposed in [139] is an improvement on the scheme presented in [15], since conspiracy attacks are prevented without attaching a random number to secret shares. The group public key is thus independent of the group size n. The protocol however requires the participants to order themselves into a ring topology, which can be limiting in some network scenarios.

Lee *et al.* [140]: The threshold-multisignature scheme in [140] also avoids conspiracy attacks without attaching a random secret to shares. The group public key is dependent on the number of group members n as the signature verifier needs the individual public values y_i of all group members and two additional parameters c_q and c_w to compute the subgroup public key, Y that is required to verifying the threshold signature. Difficulty will be experienced with this scheme when trying to eliminate the need for a trusted authority to distribute the initial group key shares.

A robust authentication mechanism is essential for securing a distributed system against *active* adversaries and central to ensuring the traceability of individual signers. As shown in Section 6.4.3, the proposed threshold-multisignature scheme uses the long-term private keys of members SK_i, provided by a public key infrastructure, to avoid conspiracy attacks even if colluding members derive or control the group secret x_Q. As a result of members including their private keys in their *individual signatures*, the public key of the scheme consists of y_Q and the public key PK_α of the subgroup α that collaborated to generate the threshold signature. The public key PK_α of the subgroup α is a function of the long-term public keys PK_i of the group members P_i, for $i \in \alpha$. Although the group public key may be perceived to be dependent on the group size n, the scheme does not introduce any additional storage requirements, since the public keys PK_i, for $i \in \alpha$ used to calculate PK_α is publicly known (traceable) and primarily required for authentication purposes.

6.4.7.2 Group-oriented signature size

The main contribution to the communication overhead, *post* signature generation, is made by the size of the threshold group signature. The threshold signature size of *threshold-multisignature* schemes is bound to be dependent on the threshold parameter or more precisely $t+c$, where $0 \leq c \leq (n-t)$. This conclusion is naturally drawn from the *traceability property* of threshold-multisignature schemes as given in Section 6.1, which specifies that any outsider must be able to retrieve the

identities of the individual signers from the threshold signature. The threshold signature must thus be bound to information explicitly linked to each of the signers that collaborated to generate the threshold signature. In the case of the proposed scheme the information is the identities of the individual signers contained in B. The individual identities of the group members can be carefully chosen to significantly reduce the size of the threshold-multisignature.

6.4.7.3 Communication cost of signature generation and verification

In terms of communication cost, the individual and threshold signature generation mechanisms of all the existing threshold-multisignature schemes and the proposed scheme are almost equivalent. Multiparty signature schemes constructed from ElGamal type (discrete logarithm based) signature variants are bound to be interactive. In round one each participant generates a commitment r_i and in the second round generates an individual signature (s_i, r_i) on an arbitrary message m. In the third round participants send their contribution to a combiner or designated clerk which constructs the threshold signature. Assume the authorized subset β of group members collaborate to sign m. This yields a three round protocols for existing schemes, which requires $(2|\beta|)$ broadcast messages and $(|\beta|)$ unicast messages. The proposed threshold-multisignature scheme is to the best of the author's knowledge the first threshold *group-oriented* signature scheme with *traceability* that allow malicious members to be positively identified and disqualified from the group. The proposed scheme also eliminates the need for a combiner. Assuming β contains at least one malicious or faulty participant, the proposed protocol will still require three rounds and only two rounds if all individual signatures satisfy Eq.(6.3). Therefore in the three round scenario the proposed protocol requires $(3|\beta|)$ broadcast messages and in the latter two round scenario only $(2|\beta|)$ broadcast messages.

6.4.7.4 Computational cost of signature generation and verification

To make a feasible comparison between the computational cost of the proposed threshold-multisignature scheme and similar schemes [15] [139] [140], it is assumed that the system parameters are chosen to yield the same time complexity for exponentiations, multiplications and summations. Although summations and in some cases multiplications contribute to an insignificant fraction of the overall time complexity, these operations are still included for the sake of completeness. Values that remain constant between different signature generations (for example the *inner parts* of the Lagrange coefficients) can be pre-computed and are therefore not included in the analysis. The computational cost of the schemes will be given in terms of the minimum threshold (t) members

(out of the total (n) group members) required to collaboratively sign an arbitrary message m.

The threshold-multisignature schemes presented in [15] [139] [140] were chosen for analysis since they all make an attempt to eliminate a conspiracy attack by t or more colluding members.

In Table 6.1 and Table 6.2 the schemes, with the assistance of a trusted key distribution center (KDC), are compared. Note that although this paper does not present a threshold-multisignature scheme with a KDC it can be trivially extended to incorporate a KDC *without* making any changes to the individual/threshold signature generation and verification procedures. Table 6.3 and Table 6.4 compares the proposed threshold-multisignature scheme, without a KDC, with the existing schemes.

In the schemes proposed by Wang *et al.* [139] and Li *et al.* [141], the public values of the individual signers are captured within the Lagrange interpolation polynomial $h(y)$ as given in Eq.(6.7). The coefficients $(b_0, b_1, \ldots, b_{t-1})$ of $h(y)$ are appended to the threshold signature by the combiner/clerk, which enable the threshold signature verifier to evaluate $h(y)$ in order to identify the individual signers. In Section 6.4.2 it was shown that appending the set of identities B to the threshold signature is a better solution, saving the threshold signature verifier the computational effort of evaluating $h(y)$. Another factor to consider is the computational cost of computing the coefficients $(b_0, b_1, \ldots, b_{t-1})$. It can be shown that the total computational cost of computing $(b_0, b_1, \ldots, b_{t-1})$ from t data pairs is given as: Multiplications: $t(\sum_{i=1}^{t-2}[\binom{t-1}{i+1}i] + (t-2) + t + 2)$, Summations: $t(\sum_{i=1}^{t-2}[\binom{t-1}{i+1} - 1] + (t-1) + t)$, Inversions: t. It should be clear that using an interpolation polynomial $h(y)$ to identify the individual signers is impractical for large values of threshold t.

The computational overhead that causes the most concern is the number of exponentiations in the individual signature verification equation (Eq.(6.3)) and threshold signature verification equation (Eq.(6.6)), which are anticipated to contribute the bulk of the verification time complexity. The justification for looking critically at the verification processes is substantiated by the notion that a signature is normally generated only once, but many times verified. The optimum number of exponentiations for an ElGamal type signature variant is 2 [53]. It can thus be concluded that the proposed threshold-multisignature scheme is superior to existing schemes since it requires only two exponentiations for threshold signature verification, while guaranteeing *break-resistance*. For individual signature verification three exponentiations are required, one more than the optimal two exponentiations. The additional exponentiation is as a consequence of satisfying the stronger break-resistance property. The proposed scheme also provides improved efficiency for all other operations, with or without the assistance of a trusted authority, which includes, individual signature

Table 6.1: Computational cost comparison of individual signature generation and verification (with the assistance of a trusted share distribution center)

Protocols	Signature generation			
	Summations	Multiplications	Exponentiations	Inversions
Li *et al.* [15]	1	(t - 1) + (t - 2) + 2	1	0
Wang *et al.* [139]	1	(t - 2) + 4	3	1
Lee *et al.* [140]	2	(t - 1) + (t - 2) + 4	1	0
PTMS* (Section 6.2.3)	2	(t - 1) + (t - 2) + 2	1	0
Protocols	Signature verification			
	Summations	Multiplications	Exponentiations	Inversions
Li *et al.* [15]	0	t((t - 2)+ 1)	3t	0
Wang *et al.* [139]	0	t	3t	0
Lee *et al.* [140]	0	t[(t - 2) + 3]	5t†	0
PTMS* (Section 6.2.4)	0	2(t - 1)	3(t - 1)	0

* Proposed threshold-multisignature scheme (PTMS), see Section 6.2.
† Note that computations of the individual signatures are performed modulo n.

generation (Table 6.1 and Table 6.3) and threshold signature generation (Table 6.2 and Table 6.4).

6.4.7.5 Efficiency of initial key distribution and key redistribution/update protocols

The proposed threshold cryptosystem is constructed with the round optimal DKG protocol presented in Chapter 6.2.2, which effectively eliminates the need for a mutually trusted key distribution center (KDC). Existing threshold-multisignature schemes that do not need the assistance of a trusted authority [15] [139] [141], do not use secure and efficient one round DKG protocols to initialize the threshold cryptosystem. These schemes also do not consider *proactive* security because they do not allow for the periodic updating of secret shares nor do they permit *dynamic* group membership since they cannot redistribute the secret shares to a new access structure. Based on the above, the existing systems are rigid concerning the threshold security parameters (n, t), which sets the security/availability tradeoff. Accordingly such systems cannot respond to changes in the networking environment. For these reasons the efficiency of the proposed threshold-multisignature scheme's system setup and maintenance procedures cannot be compared to the existing schemes.

It is the view of the author that the security of the underlying DKMI influences the security of the signature scheme to such an extent that the threshold group-oriented signature scheme cannot be considered separately from the DKMI.

Refer to Chapter 5.4.3 for an overview of the efficiency analysis of the DKMI.

Table 6.2: Computational cost comparison of threshold signature generation and verification (with the assistance of a trusted share distribution center)

Protocols	Signature generation			
	Summations	Multiplications	Exponentiations	Inversions
Li *et al.* [15]	t - 1	t - 1	0	0
Wang *et al.* [139]	s†	2(t - 1)+ m‡	0	t
Lee *et al.* [140]	t - 1	t - 1	0	0
PTMS* (Section 6.2.5)	t - 1	0	0	0

Protocols	Signature verification			
	Summations	Multiplications	Exponentiations	Inversions
Li *et al.* [15]	0	t(t - 2) + (t - 1) + 2	t + 2	0
Wang *et al.* [139]	0	n(t - 1) + 1	n(t - 2) + 3	0
Lee *et al.* [140]	0	(t - 1) + 2	5	0
PTMS*(Section 6.2.6)	0	t - 1) + 2	2	0

* Proposed threshold-multisignature scheme (PTMS), see Section 6.2.

† $s = t(\sum_{i=1}^{t-2}[\binom{t-1}{i+1} - 1] + (t-1) + t)$.

‡ $m = t(\sum_{i=1}^{t-2}[\binom{t-1}{i+1} i] + (t-2) + t + 2)$.

Table 6.3: Computational cost comparison of individual signature generation and verification (without the assistance of a trusted share distribution center)

Protocols	Signature generation			
	Summations	Multiplications	Exponentiations	Inversions
Li *et al.* [15]	0	(t - 1) + (n - t) [(t - 2) + 1] + 1	1	0
Wang *et al.* [139]	1	(n - t)(t - 2) + (n - t - 1) + 5	(n - t) + 2	1
PTMS*(Section 6.2.3)	2	(t - 1) + (t - 2) + 2	1	0

Protocols	Signature verification			
	Summations	Multiplications	Exponentiations	Inversions
Li *et al.* [15]	0	t[(t - 2) +	3t	0
	0	(n - t - 1) + 2]	3t	0
Wang *et al.* [139]	0	t	3t	0
PTMS*(Section 6.2.4)	0	2(t - 1)	3(t - 1)	0

* Proposed threshold-multisignature scheme (PTMS), see Section 6.2.

Table 6.4: Computational cost comparison of threshold signature generation and verification (without the assistance of a trusted share distribution center)

Protocols	Signature generation			
	Summations	Multiplications	Exponentiations	Inversions
Li *et al.* [15]	t - 1	t - 1	0	0
Wang *et al.* [139]	s[†]	2(t - 1) + m[‡]	0	t
PTMS[*] (Section 6.2.5)	t - 1	0	0	0

Protocols	Signature verification			
	Summations	Multiplications	Exponentiations	Inversions
Li *et al.* [15]	0	t[(t - 2) + (n - t - 1)] + (t - 1) + 2	t + 2	0
Wang *et al.* [139]	0	n(t - 1) + 1	n(t - 2) + 3	0
PTMS[*] (Section 6.2.6)	0	(t - 1) + 2	2	0

[*] Proposed threshold-multisignature scheme (PTMS), see Section 6.2.
[†] $s = t(\sum_{i=1}^{t-2}[\binom{t-1}{i+1} - 1] + (t-1) + t)$.
[‡] $m = t(\sum_{i=1}^{t-2}[\binom{t-1}{i+1} i] + (t-2) + t + 2)$.

6.5 Conclusion

Investigations within the fields of threshold *group-oriented* signature schemes, threshold *group* signature schemes, *multisignature* schemes and *threshold-multisignature* schemes resulted in explicitly defining the properties of threshold-multisignature schemes. The main aim of this paper is to introduce such a secure threshold-multisignature scheme. To reach this objective the secure and optimally efficient ElGamal type signature variant: $GES = (M.EG\text{II}.3.\sigma(1), r, s, h(m, r), 1)$, was extended to a multiparty setting to yield a threshold-multisignature scheme, which from the author's point of view is the first to provide a guaranteed *traceability property*. The proposed threshold-multisignature scheme was shown to satisfy all the specified security requirements and to fulfill the stronger *break-resistant property*. The threshold-multisignature signature scheme thus remains secure even if the threshold cryptosystem has been broken, i.e., the group secret or individual secret shares are known or controlled by an adversary. It was shown that the proposed threshold-multisignature scheme eliminates the latest attacks on similar threshold signature schemes with traceability. The efficiency analysis showed that the proposed threshold-multisignature scheme outperforms other existing schemes and is optimal in terms of exponentiations with respect to threshold signature verification and near optimal for individual signature verification, while providing break-resistance.

Chapter 7

Conclusions and Future Direction

Emerging technologies are more rapidly reaching a level of maturity where they can deliver new and innovative services to society. The combination of wireless networks and mobile computing enable these services to operate without any pre-existing or online infrastructure. Mobile ad hoc networks is a good example of a distributed communication system which enables individuals to communicate any where, any time, without prior arrangement. However, without adequate security mechanisms these technologies will not be adopted or users will be exposed to serious threats.

In this thesis we studied important security mechanisms related to distributed communication systems within the context of peer-to-peer and group communication. These mechanisms included:

- Authority-based peer-to-peer key management. (Chapter 2)
- Group key management. (Chapter 3 and 4)
- Distributed-key management. (Chapter 5)
- Threshold-multisignatures. (Chapter 6)

We grouped related mechanisms into three parts:

The first part of the thesis addressed authority-based peer-to-peer key management. We considered key management for mobile ad hoc networks since it provided us with room for innovation in a challenging system model. As we discussed in part one, unpredictable and highly dynamic network topologies emerge in mobile ad hoc networks due to the lack of online infrastructure, error-prone

wireless connectivity and node mobility. During our analysis we noted that routing failures are at the center of this dynamic topology. Given this insight we proposed a key management solution that turns the dynamic network topology into an advantage by exploiting the routing mechanism's inherent ability to recover from routing failures.

The scheme called, AuthBasedPKM integrates with the routing protocol and uses the routing control packets to relay keying material along virtual chains. The protocol breaks the difficult routing-security interdependence cycle as it does not rely on the routing protocol to set up a route. We analyzed our scheme to a large extend using simulations and confirmed that the communication and computational overhead of AuthBasedPKM has negligible impact on network performance under worst case scenarios. We showed that our scheme meets the requirement of improving its performance when the network becomes more dynamic, what we referred to as progressive robustness. Given a strong adversary model, we argued that our scheme is secure following a pragmatic approach.

The second part of the thesis addressed the problem of bootstrapping the security of group communication systems for ad hoc networks. Group communication systems (GCS') are supported by a complex architecture that includes a few basic protocols to be bootstrapped, namely unicast routing, group membership service, multicasting, group key agreement and data sharing. We applied a specific definition to the expression *bootstrap security* as follows: "the practical implementation of a set of techniques and procedures supporting the establishment and maintenance of keying material between authorized nodes, starting with no shared keying material between nodes prior to network formation and, without any assistance from an online authority".

We considered dynamic peer groups as the most likely group type suitable for ad hoc networks and discovered that group membership changes add a new dimension to the problem of contributory group key agreement. We noted that group membership changes, combined with the dynamic network topology, result in a system with little determinism. In response to the problem we proposed a scheme, called AdHocGKM that solves the problem as part of a layered GCS protocol architecture. To make the discussion on AdHocGKM as practical as possible we assumed the GCS to be Pilot [2] which includes multicast-group membership service, multicasting and data sharing protocols.

Following from part one, we discussed the impact of the dynamic group membership and network topology on bootstrapping the security GCS'. During our analysis we noted that the impact causing the most concern for security is mainly on the group membership service and contributory key agreement protocols. AdHocGKM contains three mechanisms for bootstrapping group

communication and security that mitigate this impact by "fighting fire with fire"[1].

- The first mechanism we discussed is the progressively robust AuthBasedPKM key distribution protocol proposed in part one of the thesis. The usefulness of the key distribution scheme was demonstrated by its ability to bootstrap the security of all the protocols in the GCS architecture.

- The second mechanisms is a primary-partition group membership service that bootstraps the GCS and exploits the inherent capability of PILOT [2] to implement progressive robustness. The data sharing service, based on a probabilistic quorum system, is used in an innovative way to maintain a single, full group membership *view*. During *view construction* only unicasting is used to avoid interdependencies with the multicasting protocol. The *view query and replication* protocol exploits the data sharing service to continuously replicate the *view* between members of the PILOT Storage Set (*StS*) [2] and therefore enable the membership service to maintain a consistent view.

- The third mechanisms is a progressively robust, contributory group key agreement scheme that also exploits the data sharing service and multicasting in an innovative way to reliably exchange group key shares. As contributory group key agreement schemes cannot tolerate multiple views, the scheme contains mechanisms to construct the group key in a predictable and controllable fashion for initial and auxiliary key agreement operations. The contributory group key agreement scheme is based on a comprehensive analysis of existing key agreement schemes and their applicability to ad hoc networks.

  Taking a sensible approach, we argued that the scheme is secure in a widely accepted and strong adversary model. Finally, we analyzed the performance of our scheme by considering the simulation results of AuthBasedPKM in part one and the underlying PILOT GCS [2]. We concluded that the communication and computational overhead of the scheme's core key distribution mechanism has negligible impact on network performance and discussed requirements for future analysis.

The third part of the thesis addressed the issue of sharing the power to use a cryptosystem between two or more group participants. Specific attention is given to secret sharing in a setting without any form of online authority. Our *distributed-key* management infrastructure (DKMI) provides distributed-key generation, updating and redistribution with a single protocol. Traditionally these areas have been studied as three separate problems. We showed how the protocol eliminates a

[1] This expression was used for gossip-based protocols [121] [2] to address different problems in the area of multicasting.

major problem with existing redistribution schemes; malicious or faulty protocol participants from the original access structure are positively identifiable by the existing honest members as well as the new members joining the threshold cryptosystem, which avoids repeating the secret redistribution protocol until all the verification conditions holds. We evaluated the computational and communication cost of the round optimal scheme and analyzed the security against mobile/active adversaries.

The fourth and final part of the thesis considered threshold-multisignature schemes that guarantee the signature verifier that at least a threshold t members participated in the generation of the group-oriented signature and that the identities of the signers can be easily established. Threshold-multisignature schemes have a broad application in group-oriented applications that require group member accountability. We defined the main properties of threshold-multisignature schemes and analyzed our scheme against these properties. Continuing from the security analysis on the DKMI, we showed that the scheme resists various attacks to which other similar schemes are vulnerable. Lastly we performed a comprehensive efficiency analysis and benchmarked our scheme against similar schemes to show that it outperforms other existing schemes and is optimal in terms of exponentiations with respect to threshold signature verification and near optimal for individual signature verification.

7.1 Summary of Contributions

The following is the list of original contributions of the thesis:

Key Management and Group Communication in Ad Hoc Networks

- A secure peer-to-peer (pairwise) or public key management scheme for authority-based ad hoc networks that exploits the unpredictable and dynamic network topology to the advantage of security. This included a comprehensive list of requirements for authority-based peer-to-peer key management in ad hoc networks. This scheme significantly expands on our preliminary work [12] [17] [13] and on [16] as one of the most comprehensive surveys in the field of key management for ad hoc networks.
- A secure authority-based public key establishment (APKE) protocol that allows an entity to negotiate with an *off-line* trusted authority for a public/private key pair in such a way that the authority does not learn the private key. This protocol is a solid building block for our peer-to-peer key management scheme.

- A primary-partition membership service to bootstrap group communication systems in ad hoc networks. The scheme uses an innovative approach to ensure a single, full membership view by exploiting the inherent capabilities the group communication system to implement progressive robustness based on a probabilistic quorum system [2].

- A comprehensive survey and analysis on the existing group key agreement schemes and their suitability for ad hoc networks.

- An approach for implementing secure group key management schemes in ad hoc networks that exploits changes in group membership and the unpredictable and dynamic network topology to "fight fire with fire"[2] when bootstrapping security for group communication systems. We showed how the scheme integrates into the layered protocol architecture of a popular group communication system for ad hoc networks to make the discussion practical. The discussion highlighted our design philosophy for ad hoc network security mechanisms given below.

- Based on our authority-based peer-to-peer key management scheme, group key management scheme and group membership service we defined an innovative approach to implement progressive robustness for security mechanisms in ad hoc networks. The approach exploits the unpredictable and dynamic network topology of ad hoc networks and the inherent ability of existing networking protocols to mitigate the impact of route failures caused by the lack of online infrastructure, error-prone wireless connectivity and node mobility. The result is security mechanisms that achieve even better performance as the environment becomes increasingly hostile. This approach can be used to overcome some of the main challenges of securing mobile ad hoc networks.

Distributed-Key Management and Group Signature Schemes

- We defined the notion of a distributed-key management infrastructure (DKMI) that includes distributed-key generation (DKG), distributed-key updating (DKU) and distributed-key redistribution (DKR). A DKMI is required to support a threshold cryptosystems.

- A DKMI scheme that performs DKG, DKU and DKR with a single protocol. Traditionally these schemes have been studied separately. We solved a major problem with existing verifiable secret sharing schemes to avoid repeating the secret redistribution/updating protocol until all the verification conditions holds.

[2]This expression was used for gossip-based protocols [121] [2] to address different problems in the area of multicasting.

· A threshold-multisignature scheme that guarantees the signature verifier that at least a threshold t members participated in the generation of the group-oriented signature and that the identities of the signers can be easily established. This included the first comprehensive list of requirements for threshold-multisignature schemes. Specifically we defined the novel property of *break-resistance* for threshold cryptosystems; an adversary in possession or control of the group secret key and/or the individual secret shares of any number of group members cannot generate a valid threshold-multisignature and/or partial/individual signatures.

7.2 Direction for Future Research

Our future work involves a few interesting research and development problems that need further exploration:

· Based on preliminary analysis we believe that our authority peer-to-peer key management scheme, combined with an online certificate authority (such as DICTATE [28]) is suited for vehicular networks. Further analysis is required to better understand the requirements of key management in vehicular networks and to analyze a hybrid scheme using simulations.

· Group communication systems are still an immature area in ad hoc networks with much room for improvement and should be designed to meet strong safety and liveness properties as defined by Chockler *et al* [10].

· Analyzing group communication protocols for ad hoc networks is difficult due to shortcomings in current simulation engines. Fortunately, the nature of our schemes allowed us to avoid this problem and still argue the essence of our schemes. Nevertheless, a GCS simulator is required for ad hoc networks and interesting results could be derived from its testing.

· Research is needed to design practical distributed-key and group signature schemes for dynamic peer groups in ad hoc networks. We only address these problems in conventional networks [18], but note it may be possible to implement these scheme in ad hoc networks with the mechanisms explored in this thesis.

· Finally, primary-partition membership services and contributory group key agreements for ad hoc networks are still relatively new and should be further investigated.

Bibliography

[1] M. Steiner, G. Tsudik, and M. Waidner, "Key Agreement in Dynamic Peer Groups," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 8, pp. 769–780, 2000.

[2] E. P. Luo, J. and J. Hubaux, "Pilot: Probabilistic lightweight group communication system for ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 2, pp. 164–179, 2004.

[3] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network: special issue on network security*, vol. 13, no. 6, pp. 24–30, 1999.

[4] S. Capkun, L. Buttyan, and J.-P. Hubaux, "Self-Organized Public-Key Management for Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52–64, 2003.

[5] S. Capkun, J. Hubaux, and L. Buttyan, "Mobility Helps Peer-to-Peer Security," *IEEE Transactions on Mobile Computing*, vol. 5, no. 1, pp. 43–51, 2006.

[6] J.-P. Hubaux, S. Capkun, and J. Luo, "The Security and Privacy of Smart Vehicles," *IEEE Security & Privacy Magazine*, vol. 2, no. 3, pp. 49–55, 2004.

[7] M. Raya and J. P. Hubaux, "The Security of Vehicular Ad Hoc Networks," in *proc. ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'05)*, November, 7 2005, to appear.

[8] S. Androutsellis-Theotokis and D. Spinellis, "A Survey on Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[9] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based Group Key Agreement," *ACM Transactions on Information Systems Security (TISSEC)*, vol. 7, no. 1, pp. 60 – 96, 2004.

[10] G. V. Chockler, I. Keidar, and R. Vitenberg, "Group communication specifications: a comprehensive study," *ACM Computing Surveys (CSUR)*, vol. 33, no. 4.

[11] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook in Applied Cryptography*. CRC Press, 1996.

[12] J. van der Merwe, D. Dawoud, and S. McDonald, "Fully Self-Organized Peer-to-Peer Key Management for Mobile Ad Hoc Networks," in *proc. ACM Workshop on Wireless Security (WiSe'05)*, September, 2 2005.

[13] J. van der Merwe, "Key Management in Mobile Ad Hoc Networks," M.Sc. Engineering (Electronic), University of KwaZulu-Natal (UKZN), 2005.

[14] T. P. Pedersen, "A Threshold Cryptosystem without a Trusted Party," in *proc. Advances in Cryptology - EUROCRYPT'91*, 1991.

[15] C.-M. Li, T. Hwang, and N.-Y. Lee, "Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders," in *proc. Advances in Cryptology - EUROCRYPT'94*, May, 9-12 1994.

[16] J. van der Merwe, D. Dawoud, and S. McDonald, "A Survey on Peer-to-Peer Key Management for Mobile Ad Hoc Networks," *ACM Computing Surveys (CSUR)*, vol. 39, no. 1, pp. 1–32, 2007.

[17] ——, "Key Distribution in Mobile Ad Hoc Networks based on Message Relaying," in *proc. Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS'07)*, May, 22-25 2007.

[18] ——, "A Fully Distributed Proactively Secure Threshold-Multisignature Scheme," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 562–575, 2007.

[19] J. van der Merwe and D. Dawoud, "Key Management for Dynamic Peer Groups in Mobile Ad Hoc Networks," in *Mobile Peer-to-Peer Computing for Next Generation Distributed Environments: Advancing Conceptual and Algorithmic Applications*, B.-C. Seet, Ed. IGI Global, 2009, pp. 241–281.

[20] ——, "Bootstrapping Group Communication and Security in Ad Hoc Networks," 2010, in submission.

[21] ——, "Dynamic Network Topology Advances Security in Mobile Ad Hoc Networks," 2010, in submission.

[22] J. van der Merwe, D. Dawoud, and S. McDonald, "A Public Key Management Scheme and Threshold-Multisignature Scheme for Mobile Ad Hoc Networks," *SAIEE Africa Research Journal (Transactions of SAIEE)*, vol. 97, no. 1, pp. 1–32, 2006.

[23] ——, "Trustworthy Key Management for Mobile Ad Hoc Networks," in *proc. Southern African Telecommunication Networks and Applications Conference (SATNAC'04)*, September, 6-8 2004.

[24] ——, "A Survey on Peer-to-Peer Key Management for Military Type Mobile Ad Hoc Networks," in *proc. Military Information and Communications Symposium of South Africa*, 2005.

[25] ——, "Group Key Management for Military Type Mobile Ad Hoc Networks," in *proc. Military Information and Communications Symposium of South Africa*, 2005.

[26] ——, "Public Key Management for Military Type Mobile Ad Hoc Networks," in *proc. Military Information and Communications Symposium of South Africa*, 2005.

[27] ——, "Self-Organized Peer-to-Peer Key Management for Mobile Ad Hoc Networks," in *proc. Southern African Telecommunication Networks and Applications Conference (SATNAC'05)*, 2005.

[28] J. Luo, J.-P. Hubaux, and P. T. Eugster, "DICTATE: DIstributed CerTification Authority with probabilisTic frEshness for Ad Hoc Networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 4, pp. 311–323, 2005.

[29] Y.-C. Hu, D. B. Johnson, and A. Perrig, "Ariadne: A Secure OnDemand Routing Protocol for Ad Hoc Networks," in *proc. Eighth ACM International Conf. on Mobile Computing and Networking (Mobicom'02)*, September 2002.

[30] ——, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," *Workshop on Mobile Computing Systems and Applications. IEEE*, 2002.

[31] P. Papadimitratos and Z. J. Haas, "Secure Routing for Mobile Ad Hoc Networks," in *proc. SCS Communication Network and Distributed System Modeling and Simulation Conf. (CNDS'02)*, January, 27-31 2002.

[32] M. Guerrero Zapata, "Key management and Delayed Verification for Ad hoc networks," *Journal of High Speed Networks*, vol. 15, no. 1, pp. 93–109, 2006.

[33] R. B. Bobba, L. Eschenauer, V. D. Gligor, and W. Arbaugh, "Bootstrapping Security Associations for Routing in Mobile Ad-Hoc Networks," in *proc. IEEE Global Telecommunications Conference*, December 2003.

[34] Y. Zhang, W. Lee, and Y.-A. Huang, "Intrusion Detection Techniques for Mobile Wireless Networks," *Wireless Networks*, vol. 9, no. 5, pp. 545–556, 2003.

[35] S. Yi and R. Kravets, "MOCA: Mobile certificate authority for wireless ad hoc networks," in *proc. of the 2nd Annual PKI Research Workshop (PKI 2003)*, April, 28-29 2003.

[36] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks," *IEEE Personal Communications*, vol. 8, no. 1, pp. 16–28, 2001.

[37] S. Xu, S. Capkun, and T. Beth, "Distributed and Secure Bootstrapping of Mobile Ad Hoc Networks: Framework and Constructions," *ACM Transactions on Information and System Security*, vol. 12, no. 1, pp. 1–37, 2008.

[38] S. Capkun, M. Cagalj, R. Rengaswamy, I. Tsigkogiannis, J.-P. Hubaux, and M. Srivastava, "Integrity Codes: Message Integrity Protection and Authentication Over Insecure Channels," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 4, pp. 208–223, 2008.

[39] S. Capkun, M. Cagalj, G. Karame, and N. Tippenhauer, "Integrity Regions: Authentication Through Presence in Wireless Networks," *IEEE Transactions on Mobile Computing*, 2010, to appear.

[40] C.-H. Chen, C.-W. Chen, K. Cynthia, Y.-H. Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y. Yang, and T.-C. Wu, "GAnGS: Gather, Authenticate 'n Group Securely," in *proc. 14th Annual International Conf. on Mobile Computing and Networking (MOBICOM'08)*, September, 14-19 2008.

[41] Y. Zhang, L. Zhu, and L. Feng, "Key Management and Authentication in Ad Hoc Network based on Mobile Agent," *Journal of Networks*, vol. 4, no. 6, pp. 487–494, 2009.

[42] H. Petersen and P. Horster, "Self-certified keys - Concepts and Application," in *proc. Third Conf. on Communication and Multimedia Security*, September 22-23 1997.

[43] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," in *proc. Advances in Cryptology: Crypto'84*, 1984.

[44] P. Horster, M. Michels, and H. Petersen, "Hidden signature schemes based on the discrete logarithm problem and related concepts," in *proc. Communications and Multimedia Security*, 1995.

[45] M. Girault, "Self-certified public keys," in *proc. Advances in Cryptology - EUROCRYPT'91*, 1991.

[46] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[47] G. Acs, L. Buttyan, and I. Vajda, "Provably Secure On-demand Source Routing in Mobile Ad Hoc Networks," *IEEE Trans. on Mobile Computing*, vol. 5, no. 11, pp. 1533–1546, 2006.

[48] J. Wu and F. Dai, "Efficient Broadcasting with Guaranteed Coverage in Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 3, pp. 259–270, 2005.

[49] N. Koblitz and A. J. Menezes, "Another Look at "Provable Security"," *J. of Cryptology*, vol. 20, no. 1, pp. 3 – 37, 2007.

[50] M. Cagalj, S. Capkun, and J. Hubaux, "Key agreement in peer-to-peer wireless networks," *Proceedings of the IEEE (Special Issue on Cryptography and Security)*, vol. 94, no. 2, pp. 467–478, 2006.

[51] C. P. Schnorr and M. Jakobsson, "Security of Discrete Log Cryptosystems in the Random Oracle and Generic Model," in *proc. The Mathematics of Public-Key Cryptography*, June, 12- 17 1999.

[52] ——, "Security of Signed ElGamal Encryption," in *proc. Advances in Cryptology - ASIACRYPT '00*, December,3-7 2000.

[53] P. Horster, M. Michels, and H. Petersen, "Generalized ElGamal signatures for one message block," in *proc. 2nd Int. Workshop on IT-Security*, September, 22-23 1994.

[54] G. Montenegro and C. Castelluccia, "Crypto-based Identifiers (CBIDs): Concepts and Applications," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 97–127, 2004.

[55] ——, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses," in *proc. Network and Distributed System Security Symposium (NDSS'02)*, February 2002.

[56] G. O'Shea and M. Roe, "Child-proof authentication for MIPv6 (CAM)," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 2, pp. 4–8, 2001.

[57] National Institute of Standards and Technology (NIST), "Secure Hash Standard," U.S. Department of Commence, Federal Information Processing Standards Publication FIPS PUB 180-1, April,17 1995.

[58] X. Wang, Y. L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1," in *proc. 25th Annual International Cryptology Conference (Crypto'05)*, August, 14-18 2005.

[59] M. Bellare, R. Canetti, and H. Krawczyk, "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols," in *proc. 30th Annual ACM Symposium on the Theory of Computing*, 1998.

[60] M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution," in *proc. CRYPTO*, 1993.

[61] "The Network Simulator - ns-2," available at www.isi.edu/nsnam/ns.

[62] J.-Y. Boundec Le and M. Vojnovic, "Perfect Simulation and Stationarity of a Class of Mobility Models," in *proc. IEEE Infocom*, March, 13-17 2005.

[63] W. Navidi and T. Camp, "Stationary Distributions for the Random Waypoint Mobility Model," *IEEE Transactions on Mobile Computing*, vol. 3, no. 1, pp. 99–108, 2004.

[64] C. E. Perkins, E. Belding-Royer, and S. R. Das, "Secure Ad Hoc On-demand Distance Vector (SAODV) Routing," July 2003, RFC 3561.

[65] D. Hankerson, A. J. Menezes, and S. A. Vanstone, *Guide to Elliptic Curve Cryptography*, ser. Springer Professional Computing. Springer, 2004.

[66] I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," *IEEE Transactions on Information Theory*, vol. 28, no. 5, pp. 714–720, 1982.

[67] K. Becker and U. Wille, "Communication complexity of group key distribution," in *proc. Fith ACM Conf. on Computer and Communications Security*, November 1998.

[68] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *proc. Advances in Cryptology - EUROCRYPT'94*, May, 9-12 1994.

[69] A. C.-F. Chan, "Distributed Symmetric Key Management for Mobile Ad Hoc Networks," in *proc. 23rd Conf. of the IEEE Communications Society*, March, 7-11 2004.

[70] Y. Amir, G. Ateniese, D. Hasse, Y. Kim, C. Nita-Rotaru, T. Schlossnagle, J. Schultz, J. Stanton, and G. Tsudik, "Secure Group Communication in Asynchronous Networks with Failures: Integration and Experiments," in *proc. 20th IEEE International Conf. on Distributed Computing Systems*, April 10-13 2000.

[71] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644– 654, 1976.

[72] Dutta, R and Barua, R, "Overview of Key Agreement Protocols," Tech. Rep. 2005/289, 2005, ePrint Archive, http://eprint.iacr.org.

[73] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman Key Distribution Extended to Groups," in *proc. Third ACM Conf. on Computer and Communication Security*, March 1996.

[74] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik, "On the Performance of Group Key Agreement Protocols," *ACM Transactions on Information Systems Security (TISSEC)*, vol. 7, no. 3, pp. 457 – 488, 2004, accepted for publication.

[75] G. Ateniese, M. Steiner, and G. Tsudik, "Authenticated Group Key Agreement and Friends," in *proc. 5th ACM Confernce on Computer and Communications Security*, November, 2-5 1998.

[76] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and Approaches for Distributed Sensor Security," Cryptographic Technologies Group, Trusted Information Systems, NAI Labs, Technical Report 00-010, September, 1 2000.

[77] L. Buttyan and J. P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," *ACM Mobile Networks and Applications*, vol. 8, no. 5, pp. 579–592, 2003.

[78] O. Pereira and J.-J. Quisquater, "Generic Insecurity of Cliques-Type Authenticated Group Key Agreement Protocols," in *proc. 17-th IEEE Computer Security Foundations Workshop (CSFW'04)*, June 2004.

[79] Y. Amir, Y. Kim, C. Nita-Rotaru, J. Schultz, J. Stanton, and G. Tsudik, "Secure Group Communication Using Robust Contributory Key Agreement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 5, pp. 468–480, 2004.

[80] Y. Amir, Y. Kim, C. Nita-Rotaru, J. Stanton, and G. Tsudik, "Secure Spread: An Integrated Architecture for Secure Group Communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 3, pp. 248–261, 2005.

[81] Z. J. Haas and S. Tabrizi, "On Some Challenges and Design Choices in Ad-Hoc Communications," in *proc. IEEE Military Communications Conf. (MILCOM'98)*, October, 18-21 1998.

[82] N. Asokan and P. Ginzboorg, "Key Agreement in Ad-hoc Networks," in *proc. ACM Workshop on Wireless Security (WiSe 2002)*, September, 28 2002.

[83] S. M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks," in *proc. IEEE Symposium on Research in Security and Privacy*, May 1992.

[84] E. Bresson and M. Manulis, "Securing Group Key Exchange against Strong Corruptions," in *proc. ACM Symposium on Information, Computer and Communications Security (ASI-ACCS)*, 2008.

[85] ——, "Securing Group Key Exchange against Strong Corruptions and Key Registration Attacks," *International Journal of Applied Cryptography*, vol. 8, no. 5, pp. 579–592, 2003.

[86] L.-T. Desmedt, Y and M. Burmester, "Scalable Authenticated Tree Based Group Key Exchange for Ad-Hoc Groups," *Financial Cryptography and Data Security*, vol. 4886, pp. 104–118, 2006.

[87] M. Burmester and Y. Desmedt, "Efficient and Secure Conference Key Distribution system," *Security Protocols*, vol. 1189, pp. 119–129, 2006.

[88] J. Katz and M. Yung, "Scalable Protocols for Authenticated Group Key Exchange," *Journal of Cryptology*, vol. 20, no. 1, pp. 85 – 113, 2007.

[89] M. Hietalahti, "Efficient Key Agreement for Ad Hoc Networks," Department of Computer Science and Engineering, Laboratory for Theoretical Computer Science, Helsinki University of Technology, Espoo, Finland, Master's Thesis, May, 23 2001.

[90] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, August 2002.

[91] D. G. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A Secure Audio Teleconference System," in *proc. Advances in Cryptology - CRYPTO'88*, August 1988.

[92] B.-R. I. V. Augot, D and D. Sacchetti, "A Three Round Authenticated Group Key Agreement Protocol for Ad hoc Networks," *Pervasive and Mobile Computing*, vol. 3, no. 1, pp. 36–523, 2007.

[93] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *proc. Advances in Cryptology - EUROCRYPT'94*, May, 9-12 1994.

[94] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," in *proc. IEEE Symposium of Privacy and Security*, May, 11-14 2003.

[95] L. Eschenauer and V. D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," in *proc. 9th ACM Conf. on Computer and Communication Security (ACM CCS'02)*, November, 17-21 2002.

[96] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On Data Banks and Privacy Homomorphisms," in *Foundations of Secure Computation.* Academic Press, New York, 1978, pp. 169–179.

[97] N. Ahituv, Y. Lapid, and S. Neumann, "Processing Encryted Data," *Communications of the ACM*, vol. 20, no. 9, pp. 777–780, 1987.

[98] E. F. Brickell and Y. Yacobi, "On Privacy Homomorphism," in *proc. Advances in Cryptology - EUROCRYPT'87*, 1988.

[99] P. Paillier, "Public-key Cryptosystems Based on Composite Degree Residuosity Classes," in *proc. Advances in Cryptology- EUROCRYPT'99*, 1999.

[100] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Mobile Networking for Smart Dust," in *proc. ACM/IEEE International Conf. on Mobile Computing and Networking (MobiCo'99)*, August, 17-19 1999.

[101] M. Brown, D. Cheung, D. Hankerson, J. Hernandez, M. Kirkup, and A. Menezes, "PGP in constrained wireless devices," in *proc. 9th USENIX Security Symposium*, August, 14-17 2000.

[102] W. Diffie, P. C. van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Designs Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.

[103] G. Ateniese, M. Steiner, and G. Tsudik, "New Multiparty Authentication Services and Key Agreement Protocols," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 628–639, 2000.

[104] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.

[105] T. D. Chandra, V. Hadzilacos, S. Toueg, and B. Charron-Bost, "On the impossibility of group membership," in *proc. 15th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, May 1996.

[106] Y. Amir and J. Stanton, "The Spread Wide Area Group Communication System," Department of Computer Science, George Washington University, Technical Report CNDS-98-4, 1998.

[107] E. P. Luo, J and J. Hubaux, "Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks," in *proc. 122nd IEEE INFOCOM*, March, 30 2003.

[108] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad-hoc Wireless Networks," in *Mobile Computing*, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996, pp. 153–181.

[109] C. E. Perkins and E. M. Belding-Royer, "Ad-hoc On-demand Distance Vector Routing," in *proc. The Second IEEE Workshop on Mobile Computing Systems and Applications (IEEE WMCSA'99)*, February 1999.

[110] H. J. Luo, J and P. Eugster, "PAN: Providing Reliable Storage in Mobile Ad Hoc Networks with Probabilistic Quorum Systems," in *proc. 4th ACM International Symposium on Mobile Ad Hoc networking & Computing (MobiHoc'03)*, June, 1-3, 2003.

[111] Y. Amir, "Replication using group communication over a partitioned network," Institute of Computer Science, The Hebrew University of Jerusalem, Ph.D. dissertation, 1995.

[112] Z. J. Haas and B. Liang, "Ad Hoc Mobility Management with Uniform Quorum Systems," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 228–240, 1999.

[113] M.-S. Bouassida, I. Chrisment, and O. Festor, "Group Key Management in MANETs," *International Journal of Network Security*, vol. 6, no. 1, p. 6779, January 2008.

[114] L. Liao and M. Manulis, "Tree-based Group Key Agreement Framework for Mobile Ad-Hoc Networks," *Future Generation Computer Systems*, vol. 23, no. 6, pp. 787–803, 2007.

[115] M. Manulis, "Contributory Group Key Agreement Protocols, Revisited for Mobile Ad-Hoc Groups," in *proc. 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2005)*, 2005.

[116] B. Wu, J. Wu, and Y. Dong, "An efficient group key management scheme for mobile ad hoc networks," *International Journal of Security and Networks*, vol. 4, no. 1/2, pp. 125–134, 2009.

[117] L. Briesemeister and G. Hommel, "Localized Group Membership Service for Ad Hoc Networks," in *proc. International Conference on Parallel Processing Workshops (ICPPW'02), month =*.

[118] L. Briesemeister, "Group Membership and Communication in Highly Mobile Ad Hoc Networks," PhD thesis, School of Electrical Engineering and Computer Science, Technical University of Berlin, 2001.

[119] K. Singh, A. Nedos, and S. Clarke, "TransMAN: A Group Communication System for MANETs," in *in proc. 8th International Conference on Distributed Computing and Networking (ICDCN) 06*, December, 27-30 2006.

[120] P. Mohapatra, C. Gui, and J. Li, "Group Communications in Mobile Ad Hoc Networks," *Computer*, vol. 37, no. 2, pp. 52–59, 2004.

[121] I. Gupta, K. P. Birman, and R. van Renesse, "Fighting Fire with Fire: Using Randomized Gossip to Combat Stochastic Scalability Limits," *Journal Quality and Reliability Engineering International*, vol. 18, no. 3, pp. 165–184, 2002.

[122] J. Katz and M. Yung, "Scalable Protocols for Authenticated Group Key Exchange," in *proc. Advances in Cryptology - CRYPTO'03*, August, 17-21 2003.

[123] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater, "Provably Authenticated Group Diffie-Hellman Key Exchange," in *proc. 8th ACM Conf. on Computer and Communications Security*, November, 6-8 2001.

[124] Y. Desmedt, "Society and group oriented cryptography: a new concept," in *proc. Advances in Cryptology – Crypto '87*, 1988.

[125] ——, "Threshold Cryptography," *European Transactions on Telecommunications*, vol. 5, no. 4, pp. 449–457, 1994.

[126] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems," in *proc. Advances in Cryptology - EUROCRYPT'99*, May, 2-6 1999.

[127] M. Stadler, "Publicly Verifiable Secret Sharing," in *proc. Advances in Cryptology - EUROCRYPT'96*, 1996.

[128] R. Gennaro, S. Jarecki, and H. Krawczyk, "Revisiting the Distributed Key Generation for Discrete-Log Based Cryptosystems," in *proc. RSA Security Conf. – Crypto Track*, April 2003.

[129] R. Gennaro, S. Jarecki, H. Krawczyk, and M. Rabin, T, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems," *Journal of Cryptology*, vol. 20, no. 1, pp. 51–83, 2006.

[130] P.-A. Fouque and J. Stern, "One Round Threshold Discrete-Log Key Generation without Private Channels," in *proc. Public Key Cryptography - PKC'01*, February, 13-15 2001.

[131] R. Zhang and H. Imai, "Round Optimal Distributed Key Generation of Threshold Cryptosystem Based on Discrete Logarithm Problem," in *proc. Applied Cryptography and Network Security (ACNS'03)*, October, 16-19 2003.

[132] A. Herzberg, S. Jaracki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing Or: How to Cope With Perpetual Leakage," in *proc. Advances in Cryptology - CRYPTO '95*, 1995.

[133] Y. Desmedt and S. Jajodia, "Redistributing Secret Shares to New Access Structures and Its Applications," Department of Information and Software Engineering, School of Information Technology and Engineering, George Mason University, Technical Report ISSE-TR-97-01, July 1997.

[134] T. M. Wong, C. Wang, and J. M. Wing, "Verifiable Secret Redistribution for Archive System," in *proc. First International IEEE Security in Storage Workshop*, December, 11 2002.

[135] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[136] P. Feldman, "A Practical Scheme for Non-Interactive Veriable Secret Sharing," in *proc. 28th Annual Symposium on the Foundations of Computer Science (FOCS'87)*, October, 12-14 1987.

[137] J. Camenisch and I. Damgrd, "Verifiable Encryption, Group Encryption, and their Applications to Separable Group Signatures and Signature Sharing Schemes," in *proc. Advances in Cryptology - ASIACRYPT'00*, 2000.

[138] A. Boldyreva, "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme," in *proc. Public Key Cryptography - PKC'03*, 2003.

[139] C.-T. Wang, C.-H. Lin, and C.-C. Chang, "Threshold signature schemes with traceable signers in group communications," *Computer Communications*, vol. 21, no. 8, pp. 771–776, 1998.

[140] W.-B. Lee and C.-C. Chang, "(t, n) Threshold Digital Signature with Traceability Property," *Journal of Information Science and Engineering*, vol. 15, no. 5, pp. 669–678, 1999.

[141] Z.-C. Li, J.-M. Zhang, J. Luo, W. Song, and Y.-Q. Dai, "Group-oriented (t, n) threshold digital signature schemes with traceable signers," in *proc. Topics in Electronic Commerce, Second International Symposium, ISEC 2001*, April, 26-28 2001.

[142] L. Harn and Y. Xu, "Design of generalised ElGamal type digital signature schemes based on discrete logarithms." *Electronics Letters*, vol. 30, no. 24, pp. 2025–2026, 1994.

[143] G. Wang, "On the security of the Li-Hwang-Lee-Tsai threshold group signatures scheme," in *proc. Fifth International Conf. on Information Security and Cryptography (ICISC'02)*, November, 28-29 2002.

[144] H. Pedersen, "How to convert any digital signature scheme into a group signature scheme," in *proc. 5th International Workshop on Security Protocols*, April, 7-9 1997.

[145] Y.-M. Tseng and J.-K. Jan, "Attacks on threshold signature schemes with traceable signers," *Information Processing Letters*, vol. 71, no. 1, pp. 1–4, 1999.

[146] T.-S. Wu and C.-L. Hsu, "Cryptanalysis of group-oriented (t, n) threshold digital signature schemes with traceable signers," *Computer Standards and Interfaces*, vol. 26, no. 5, pp. 477–481, 2004.

[147] G. Wang, X. Han, and B. Zhu, "On the Security of Two Threshold Signature Schemes with Traceable Signers," in *proc. Applied Cryptography and Network Security, First International Conf., ACNS 2003*, October, 16-19 2003.

[148] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Adaptive Security for Threshold Cryptosystems," in *proc. Advances in Cryptology - CRPYTO'99*, August, 15-19 1999.

[149] M. Michels and P. Horster, "On the Risk of Disruption in Several Multiparty Signature Schemes," in *proc. Advances in Cryptology - ASIACRYPT '96*, November, 3-7 1996.

Curriculum Vitae

Johann van der Merwe

Born in Bloemfontein, South Africa on September 2, 1981
Nationality: South African

Biography

Johann van der Merwe is in the employ of PricewaterhouseCoopers, South Africa where he is the competency leader for Security Advisory across all information security competencies. He is a security architect and specialises in Infrastructure & Network Security, Identity & Access Management and Information Security Management. Prior to joining PricewaterhouseCoopers he worked as a manager in the Security & Privacy Services team of Deloitte, South Africa. He was the service line leader for Infrastructure & Operations Security and Identity & Access Management.

Johann received his Master of Science in Electronic Engineering degree from the University of KwaZulu-Natal and was supported by ARMSCOR, the Armaments Corporation of South Africa. He is pursuing a *part-time* PhD degree in security for mobile wireless networks.

Education

- University of KwaZulu-Natal, South Africa, **PhD in Electronic Engineering**, *Part-time*, Expected: Dec. 2010
  Thesis title: *Security Mechanisms for Distributed Communication Systems*
  Thesis advisors: *Prof. Dawoud Dawoud*
- University of KwaZulu-Natal, South Africa, **MSc in Electronic Engineering**, Jan. 2000 - Dec. 2003
  Thesis title: *Key Management in Mobile Ad Hoc Networks*
  Thesis advisors: *Prof. Dawoud Dawoud* and *Mr. Stephen McDonald*

· University of Natal, South Africa, **BSc in Electronic Engineering**, Jan. 2000 - Dec. 2003

Publications

Journals and book chapter

· J. van der Merwe, D. Dawoud, and S. McDonald, A Fully Distributed Proactively Secure Threshold-Multisignature Scheme, IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 4, pp. 562-575, 2007.

· J. van der Merwe and D. Dawoud, Key Management for Dynamic Peer Groups in Mobile Ad Hoc Networks, in Mobile Peer-to-Peer Computing for Next Generation Distributed Environments: Advancing Conceptual and Algorithmic Applications, B.-C. Seet, Ed. IGI Global, 2009, pp. 241-281.

· J. van der Merwe, D. Dawoud, and S. McDonald, A Survey on Peer-to-Peer Key Management for Mobile Ad Hoc Networks, ACM Computing Surveys (CSUR), vol. 39, no. 1, pp. 132, 2007.

· J. van der Merwe, D. Dawoud, and S. McDonald, A Public Key Management Scheme and Threshold-Multisignature Scheme for Mobile Ad Hoc Networks, SAIEE Africa Research Journal (Transactions of SAIEE), vol. 97, no. 1, pp. 132, 2006.

Submitted for review

· J. van der Merwe and D. Dawoud, Bootstrapping Group Communication and Security in Ad Hoc Networks, 2010, in submission.

· J. van der Merwe and D. Dawoud, Dynamic Network Topology Advances Security in Mobile Ad Hoc Networks, 2010, in submission.

Refereed conference and workshop papers

· J. van der Merwe, D. Dawoud, and R. Peplow, Vulnerability Windows in Vehicular Communications, in proc. Wireless Communication Society, Vehicular Technology, Information Theory and Aerospace & Electronics Systems Technology (Wireless VITAE'09), Aalborg, Denmark, May 17-20, 2009.

· D. Dawoud, R. Peplow and J. van der Merwe, Ensuring Privacy in Vehicular Communication, in proc. Wireless Communication Society, Vehicular Technology, Information Theory and Aerospace & Electronics Systems Technology (Wireless VITAE'09), Aalborg, Denmark, May 17-20, 2009.

· J. van der Merwe, and D. Dawoud Key Distribution in Mobile Ad Hoc Networks based on Message Relaying, in proc. Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS07), May, 22-25 2007.

· J. van der Merwe, D. Dawoud, and S. McDonald, Fully Self-Organized Peer-to-Peer Key Management for Mobile Ad Hoc Networks, in proc. ACM Workshop on Wireless Security (WiSe05), September, 2 2005.

· J. van der Merwe, D. Dawoud, and S. McDonald, Self-Organized Peer-to-Peer Key Management for Mobile Ad Hoc Networks, in proc. Southern African Telecommunication Networks and Applications Conference (SATNAC05), 2005.

· J. van der Merwe, D. Dawoud, and S. McDonald, A Survey on Peer-to-Peer Key Management for Military Type Mobile Ad Hoc Networks, in proc. Military Information and Communications Symposium of South Africa, 2005.

· J. van der Merwe, D. Dawoud, and S. McDonald, Group Key Management for Military Type Mobile Ad Hoc Networks, in proc. Military Information and Communications Symposium of South Africa, 2005.

· J. van der Merwe, D. Dawoud, and S. McDonald, Public Key Management for Military Type Mobile Ad Hoc Networks, in proc. Military Information and Communications Symposium of South Africa, 2005.

· J. van der Merwe, D. Dawoud, and S. McDonald, Trustworthy Key Management for Mobile Ad Hoc Networks, in proc. Southern African Telecommuni-

cation Networks and Applications Conference (SATNAC04), September, 6-8 2004.

MSc Thesis

· J. van der Merwe, Key Management in Mobile Ad Hoc Networks, M.Sc. in Engineering (Electronic), University of KwaZulu-Natal (UKZN), 2005.

Professional Experience

· Information Security Competency Leader and Manager, PricewaterhouseCoopers, Johannesburg, South Africa, Dec. 2008 - current.

· Information Security Manager and Service Line Leader, Deloitte, Johannesburg, South Africa Jul. 2006 - Nov. 2008.

· Information Security Researcher and PhD degree candidate, CSIR, Pretoria, South Africa Jan. 2006 - Jun. 2006 (under contract).

· Information Security Researcher and MSc degree candidate, ARMSCOR, Durban, South Africa Jan. 2004 - Dec. 2005.

Professional Activities

· Chapter Agent, Information Security Forum (ISF), South Africa, Jan. 2009 - current.

· Member, SABS SC71F Information Security Standard Committee, Feb. 2009 - current.

· Convenor, SABS SC71F Information Security Standard Committee, Work Group 1, Information Security Management Systems, Aug. 2009 - current.

· Member, SABS SC71F Information Security Standard Committee, Work Group 5, Privacy, Mar. 2009 - current.

· Member ISG Africa, Jan. 2006 - current.

· Member ISACA, South Africa, , Apr. 2010 - current.

Security presentations and conference involvement

· Information Security South Africa (ISSA) Conference 2010 [Guest Speaker].

· ITWeb Security Summit 2010 [Chair and Member of Technical Programme Committee].

· ITWeb Mobilebiz Conference 2010 [Speaker].

· ITWeb Cloud Computing & Virtualization Conference 2010 [Speaker].

· ITWeb Service Oriented Architecture Conference [Speaker and Chair].

· ITWeb Security Summit, 2009 [Speaker and Chair].

· ISACA Chapter meeting, 2009 [Speaker].

· ISF Chapter meeting, 2009 [Speaker and Agent].

· ITWeb Enterprise Virtualisation Conference 2008 [Speaker].

· Fourth European Workshop on security and privacy in ad hoc and sensor networks (ESAS) held in the UK, 2007 [Speaker, Accepted Research Paper].

· ITWeb Security Summit 2007 [Speaker].

· ITWeb Enterprise Risk Management Conference 2007 [Speaker].

· ITWeb Mobile and Wireless Conference 2006 [Speaker].

· ACM Workshop on Wireless Security (WiSe) held in Germany, in 2005.

· SATNAC, in 2004 and 2005 [Speaker, Accepted Research Paper].

· MICSA, in 2005 [Speaker, Accepted Research Paper].