Université d'Ottawa • University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DE ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Ahmed ABDEL-HAFEZ

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Ph.D. (Electrical Engineering)

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Is the Nap Zone Controlled by a Light Sensitive Circadian Arousal Process?

A. Miri

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

L. Barbosa

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

N. Georganas

A. Hasan

D. Makrakis

C. Huang

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Secure and Efficient Group Key Agreement Protocols

Ahmed Abdel-Hafez

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the Ph.D degree in Electrical Engineering

School of Information Technology and Engineering
Ottawa-Carleton Institute of Electrical and Computer Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 0-612-89987-X
Our file Notre référence
ISBN: 0-612-89987-X

Abstract

Due to the increased popularity of group-oriented applications and protocols, securing group communication has become a critical networking issue and has received much attention in recent years. A secure and efficient group key management protocol presents one of the most fundamental challenges in group communication security. While key transport protocols may be appropriate for key establishment in one-to-many applications, many collaborative applications require distributed key agreement protocols. Proposals for key agreement protocols that have been published so far do not scale for large size groups. Moreover, the security of most of these protocols has been flawed. This thesis considers the problem of group key agreement protocols from two points of view: efficiency and security. Regarding the efficiency, a novel framework based on the extension of the Diffie-Hellman key exchange protocol is proposed. The efficiency of this framework comes from the distribution of the group members into size-bounded clusters. Using clustering improves the protocol efficiency (in both communication and computation costs), provides fault-tolerance and supports heterogenous systems. A modification to these protocols has been presented to address the problem of key agreement within ad hoc wireless networks. Regarding protocol security, a new approach for key agreement protocols analysis is proposed. To demonstrate the approach, a formal analysis of one of the existing key agreement protocols has been presented where some prac-

tical attacks have been identified. It has been shown that the possibility of most of these attacks is due to the use of improper techniques to supply key authentication. Different solutions to address these attacks have been proposed. These solutions provide authenticity and are implemented within the steps required by the key agreement protocol. These authenticated protocols are based on an entanglement of both the ephemeral key and the long-term key to provide authenticated key agreement protocols.

To my Parents,

To my wife, and

To my children, Ali and Farah

Acknowledgment

I am indebted to and enormously grateful to my supervisor, Dr. Ali Miri, for his selfless dedication to the development of this work. His assistance and contributions to every stage of the research process make this dissertation possible and taught me a great deal. Thank you.

My gratitude to my co-supervisor Dr. Luis Orozco-Barbosa, similarly, knows no bounds. His guidance, support, and direction through the development of this work have been invaluable.

I appreciate the kindness and good nature shown by all of my colleagues in the CLiCC lab at the university of Ottawa, especially Mathew Samuel who spent a lot of his time helping me during the writing-up phase of my thesis.

I am also grateful to Dr. Jean-Yves Chouinard for pushing my thinking and leading my first steps in research.

Several referees provided many useful suggestions about selected parts of this thesis material. I am also very grateful to my dissertation committee Dr. Nicolas D. Georganas, Dr. Dimitrios Makrakis and Dr. Changcheng Huang for their involvement and feedback.

As always, my family offered me a continuous support, each of them in their own way. More particularly, my wife for her understanding, assistance, patience and taking care of my children.

Contents

List of Figures IX

1 Introduction and Overview 1

1.1 Thesis Outline 4

1.2 Results 5

2 Group Key Agreement Features 8

2.1 Introduction 8

2.2 Authentication and Key Agreement Protocols 12

2.3 Two-Party Key Agreement Protocols 13

2.3.1 Diffie-Hellman Key Exchange Protocol 16

2.4 Group Key Agreement 20

2.4.1 Group Communications Security Challenges 21

2.4.2 Fault-Tolerance 23

2.4.3 Management of Groups 24

2.4.4 Handling Membership Changes 24

2.4.5 Complexity Measures and Metrics 28

2.5 Related Work 31

2.6 Summary 36

3 Efficient Group Key Agreement Protocols **38**

3.1 Introduction 38

3.2 Clustering Techniques 39

3.2.1 Desirable Attributes of the Clustering Scheme 41

3.3 Construction of the Hierarchical Structure 43

3.4 Hybrid Group Key Management (HGKM) 46

3.4.1 Initial Hybrid Group Key Management (I-HGKM) 47

3.4.2 Group Membership Changes 51

3.4.3 Discussion 59

3.5 Cluster Group Key Agreement 61

3.5.1 Initial Session Key Generation 61

3.5.2 Group Membership Changes 67

3.6 Complexity Analysis 73

3.7 Similarities With Other Approaches 75

3.8 Summary 77

4 Group Key Agreement for Ad Hoc Networks **79**

4.1 Introduction 79

4.2 Attributes of Wireless Communication Systems 81

4.3 Threats and Vulnerabilities in Wireless Networks 82

4.4 Ad Hoc Networks Security 84

4.5 Dynamic Topology GKA Protocol (DTGKA) 85

4.5.1 Initial Setting 85

4.5.2 Clusters Construction Phase 86

4.5.3 Key Agreement Phase 89

4.5.4 Security Analysis 92

4.5.5 Performance Analysis 93

4.6 Authenticated DTGKA (A-DTGKA) 94
4.7 Summary 98

5 Formal Security Analysis **100**
5.1 Introduction 100
5.2 Properties of Group Key Agreement Protocols 102
5.3 Survey of Provable Security Techniques 106
5.4 New Approach to the Analysis of GDH Protocols 108
5.5 Modelling of GDH Protocols 111
5.5.1 Protocol Description 111
5.5.2 Intruder Capabilities 114
5.5.3 Security Check 114
5.6 Analysis of Cliques Suite 115
5.6.1 Implicit Key Authentication 115
5.6.2 Perfect Forward Secrecy 120
5.6.3 Resistance to Known-Keys Attacks 122
5.7 Discussion 123
5.8 Summary 124

6 Authenticated Group Key Agreement **125**
6.1 Introduction 125
6.2 Initial Setting 126
6.3 Group MTI Key Agreement Protocol (GMTI) 128
6.3.1 Discussion 129
6.4 Group MQV Protocol (GMQV) 131
6.4.1 Protocol Description 131
6.4.2 Discussion 133

6.5 Remarks 134
6.6 Authenticated BD Protocol (ABD) 135
6.6.1 Discussion 137
6.7 Summary 139

7 Conclusion and Future Work **140**

References **144**

A Diffie-Hellman Problem **154**
A.1 The Discrete Logarithm Problem 155
A.2 The Diffie-Hellman Problem 156
A.3 DLP on Elliptic Curves 156

B IBE and Weil Pairing **158**
B.1 Identity Based Encryption: 158
B.2 The Weil Pairing: 159
B.3 Bilinearity Property 159

List of Figures

3.1 Clustering Concepts 41

3.2 Members Distribution 45

3.3 HGKM Hierarchical Scheme 48

3.4 The Hierarchy Updating after the Departure of Member $M_{1,3}$ 55

3.5 Key Regeneration After the Departure of $M_{1,3}$ 56

3.6 Hierarchical Structure of Group Members 62

4.1 Distribution of Group Members 89

List of Tables

2.1 MTI Key Agreement Protocols 19

3.1 Initial Key Generation Comparison 75
3.2 Join Protocol Comparison 76
3.3 Leave Protocol Comparison 76

List of Symbols

$\mathcal{M}$	The set of group members.
n	The cardinality of $\mathcal{M}$.
d	The maximum cluster size.
l	The maximum number of levels in the hierarchical scheme.
N_C	The number of clusters in the hierarchical scheme.
i, v, j	Indices of levels, clusters and members.
i	$1, 2, \ldots l$.
v	$1, 2, \ldots d(d-1)^{i-2}$.
j	$1, \ldots, d-1$.
$C_{i,v}$	The v^{th} cluster in the i^{th} level.
$K_{i,v}$	The session key of the cluster $C_{i,v}$.
$M_{i,v}^{j}$	The j^{th} member in the v^{th} cluster in the i^{th} level.
$\{M_{i,v}^{j}\}_s$	The descendants of member $M_{i,v}^{j}$.
M_j	The j^{th} member in $\mathcal{M}$.
$\mathbb{F}_q$	The finite field with q elements.
$\mathbb{Z}_p^*$	The multiplicative group $\mathbb{Z}_p^* = \{1, 2, ..., p-1\}$.
$r_j \in \mathbb{Z}_p^*$	The random secret contributed by M_j.

$s_j \in \mathbb{Z}_p^*$	The long-term secret key of M_j.
P_j	The long-term public key of M_j.
$\mathbb{G}$	An abelian cyclic group.
$\alpha \in \mathbb{G}$	The public generator of $\mathbb{G}$.
$E(\mathbb{Z}_p)$	The elliptic curve defined over $\mathbb{Z}_p$.
$Q \in E(\mathbb{Z}_p)$	The base point.
$\mathcal{O} \in E(\mathbb{Z}_p)$	the point at infinity.
$\mathcal{H}(m)$	The Hash function of the message m.
$\mathbf{E}_k(m)$	The symmetric encryption of message m using k as a symmetric key.
K_{ij}	The long-term key shared by M_i and M_j.
$S_n(M_i)$	The session key calculated by M_i.
$\mathbf{R}$	The Key Agreement Protocol.
R	The total number of rounds in $\mathbf{R}$.
M_s	The maximum message size.
Ex_j	The number of exponentiations/member, M_j.
Ex_T	The total number of exponentiations in the protocol.
Ex_C	The number of exponentiations/cluster.

List of Acronyms

A-DTGKA	Authenticated-Dynamic Topology Group Key Agreement.
AkA	Auxiliary Key Agreement protocols.
CDH	Computational Diffie-Hellman problem.
CGKM	Cluster Group Key Management Protocol.
CA	Certificate Authority.
DDF	Decisional Diffie-Hellman problem.
DH	Diffie-Hellman Key Exchange Protocol.
DHP	Diffie-Hellman Problem.
DLP	Discrete Logarithm Problem.
DTGKA	Dynamic Topology Group Key Agreement.
GDH	Group Diffie-Hellman.
HGKA	Hybrid Group Key Agreement Protocol.
IKA	Initial Key Agreement protocol.
KGC	Key Generation Center.
KKA	Known-Key Attack.
PFS	Perfect Forward Secrecy.
NA	Network Administrator.
PDA	Personal Digital Assistance.
TGDH	Tree Group Diffie-Hellman.
TTP	Trusted Third Party.
QoS	Quality of Service.

Chapter 1

Introduction and Overview

This chapter gives an overview of the research area. In particular, it provides a statement of the problems which will be tackled throughout the thesis. Additionally, it provides a summary of the major results and the outline of the thesis contents.

With multicast or group communications, a message is transmitted to all members of a group. Transmission savings are clearly achieved since only one transmission is needed to reach all members instead of n transmissions to n members as in the case of unicast, or point-to-point communications. Current multicast standards lack mechanisms to protect against eavesdropping, impersonation and session hijacking, thus leaving multicast open to severe threats of data manipulation and theft of service or content.

Group communication-based applications can not have their transmission protected by a firewall or any other central mechanism. This is because the communicating members are distributed across the network.

One alternative is to use public-key cryptography to secure all group messages. As a rule of thumb, public-key cryptographic algorithms are computationally around

1000 times slower than private-key cryptographic algorithms, making it prohibitively expensive. Moreover, to secure communication between a group of n members, using these techniques, each member should store (or have access to) $n-1$ authentic public keys. Furthermore, if any member wants to send a secure message to any subset of the group (or possibly the whole group), he must encrypt and send as many messages as the number of the recipients; this results in large communication and computation overhead. Another alternative is to use private-key encryption by using one of two approaches: a pairwise shared key or a common group key. Using a pairwise shared key is not an efficient way in terms of storage and number of encryption processes and transmission required as in the case of public-key cryptography. The second option of using a common shared key, seems to provide the most suitable solution.

The use of private-key cryptography, based on a common group key, requires the generation and distribution of that common key to all authorized members of the group. This process is known in literature as the *key establishment protocol*. Assuming that every member agrees on the same key, all group messages can be encrypted using this shared key. If we presume that the intruder has no way to retrieve the group key then group members are protected. Thus one of the main design challenges, in secure and reliable group communication systems, is to establish a common group key that is available to all authorized members in an efficient and secure way. Two types of key establishment protocols have been defined in literature [52]: Key transport and key agreement protocols. While key transport protocols may be appropriate for key establishment in one-to-many applications, many collaborative applications require distributed key agreement protocols.

Generally, group communication security protocols are multi-round protocols. Consequently, protocol efficiency is a major concern due to the direct correlation of the number of the participants to the computation and communication complex-

ities, which has not been properly addressed by the previously proposed protocols. Accordingly, one of the first questions that needs to be addressed is: how to design an efficient group key agreement protocol or improve the *efficiency* of the existing protocols?

Most of existing protocols so far have been focused on extending the well-known 2-party Diffie-Hellman key exchange protocol (DH) [28] to operate as a multi-party (group) key agreement protocol. The security of these protocols has been based on the assumption that the intractability of the group Diffie-Hellman problem can be reduced to the intractability of 2-party Diffie-Hellman. Contrary to this assumption, group key agreement is not a simple extension of two-party key agreement protocols. In fact, there are many important differences and several design and implementation challenges that have to be dealt with [21]. These differences stem from the differences in nature of point-to-point communications and group communications. A distinct definition of security requirements is needed which leads to the following question: how to design a *secure* group key agreement protocol?

This thesis focuses on the efficiency and security of group key agreement protocols within a group of members with a highly dynamic membership nature.

Key agreement protocols can be considered as the cornerstone of group services. Generally, efficiency and security are two natural but conflicting goals in cryptography. As a result, there is always a trade off between them, which should be based on the required degree of security and the capability of protocol users. A common approach considers that all group members have a symmetric relationship and are treated equally[1]. In particular, roles such as group leadership are not fixed

[1]This statement may not be valid when considering a heterogenous systems like incorporation of mobile devices or receive only members. In this case, leadership choosing is controlled by a system policy.

beforehand. Moreover, there is no central authority with more control than other members and assignment of controllership roles should be only a matter of policy and orthogonal to the main key management protocol.

1.1 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 introduces a survey of the key agrement protocols and their related issues. In particular, a discussion of the differences and requirements related to group key agreement protocols will be presented. Chapter 3 presents the framework for the scalable key agreement protocols. This framework adopts the idea of clustering, which will also be discussed. Two different key agreement protocols (HGKM, CGKA) based on members clustering are then described and their complexities are analyzed. Due to the different nature of ad hoc wireless networks, the fixed structure used in these two protocols is not suitable for these kinds of networks. In Chapter 4, a modified version (DTGKA) is presented to accommodate the special nature of ad hoc wireless networks. The security of the framework mentioned in Chapter 3 is based on the security of the building block of each cluster. In Chapter 5, a formal model of the GDH family will be presented. This family includes the protocols used as a building block of the proposed framework. A detailed and rigorous proof for this family of protocols will also be presented. The formal analysis used for this family points out some practical attacks against the existing protocols. In Chapter 6, counterattack solutions to these weaknesses will be given. These solutions will be adopted to provide building blocks for authenticated key agreement protocols. Conclusions and a discussion of future work are given in Chapter 7.

1.2 Results

The major results of this thesis can be summarized as follows:

1. An efficient solution to group key agreement protocols based on the extension of two-part Diffie-Hellman key exchange protocol: In this solution clustering techniques, which are commonly used by most group communication applications will be adopted. After distribution of the group members into clusters, two different protocols are presented:

   - Hybrid Group Key Management (HGKM): In this approach both key agreement and key transport protocols are adapted to constitute an efficient framework to generate/regenerate the group session key.
   - Cluster Group Key Agreement (CGKA): The CGKA protocol offers an alternative to the HGKM distributed key transfer protocol, in which the intermediate cluster's session keys are never transmitted through the network and can provide a multilevel security environment.

2. An Authenticated Group Key Agreement Protocols for Ad hoc Networks: This is a modified version of the CGKA protocol, which will provide a secure key management solution for ad hoc wireless networks. This modified solution conforms to the characteristics of typical ad hoc wireless networks.

3. A new formal security analysis of a particular class of authenticated group key agreement protocols (GDH): This analysis will involve the identification of some practical attacks. The extension of this analysis to HGKM and CGKA will be also provided.

4. A novel authenticated group key agreement protocol: The new authenticated protocol will use an entanglement of both long-term secret key and ephemeral

secret key for group members. This will provide an authenticated key agreement protocol and deal with the discovered attacks. An important feature of this approach is that the authentication is fulfilled as part of the group key agreement protocol and not in a separate step

Most of above results are under consideration for possible publication in various conferences and journals.

1. A. Abdel-Hafez, A. Miri, and L. Orozco-Barbosa. "Hybrid Key Management for Group Communication". *In the Proceedings of The Eighth Canadian Workshop on Information Theory CWIT* '03, pp. 71 – 74, May 2003.

2. A. Abdel-Hafez, A. Miri, and L. Orozco-Barbosa. "Scalable Key Agreement Protocol for Group Communication". *Submitted to the ACM/IEEE Transactions on Networking.*

3. A. Abdel-Hafez, A. Miri, and L. Orozco-Barbosa. "Scalable and Fault-Tolerant Key Agreement Protocol for Dynamic Groups". *Submitted to the Journal of Computer Communications.*

4. A. Abdel-Hafez, A. Miri, and L. Orozco-Barbosa. "Authenticated Secure Communications in Wireless Networks" *Submitted to The Fifth European Wireless Conference (EW*2004*)*

5. A. Abdel-Hafez, A. Miri, and L. Orozco-Barbosa. "Formal Security Analysis of GDH Key Agreement Protocols" *Submitted to the IASTED International Conference on Communication, Network, and Information Security (CNIS 2003)*.

6. A. Abdel-Hafez, A. Miri, and L. Orozco-Barbosa. "Authenticated Group Key Agreement Protocols for Ad hoc Wireless Networks" *Preprint.*

7. A. Abdel-Hafez, A. Miri, and L. Orozco-Barbosa. "Secure and Efficient Key Agreement protocols for Ad hoc Wireless Networks" *To be submitted to the Ninth IEEE Symposium on Computers and Communications (ISCC'2004)*

Chapter 2

Group Key Agreement Features

In this chapter, various dimensions of key agreement, in the context of group communications, are discussed. In particular, the main security requirements for two-party key agreement protocols and the additional security requirements for group key agreement protocols are discussed. This is followed by a discussion of some group communications related topics, such as fault tolerance, group management and complexity measures. Finally, a brief survey of the previously proposed group key agreement protocols and a discussion of their characteristics are presented.

2.1 Introduction

With the widespread use of the Internet, the popularity of *group communication*-based applications has grown considerably. Group communication is a means of providing multi-point to multi-point communications by organizing processes in groups. Current group-oriented applications include live multi-party conferences, on-line video games, collaborative workspaces, remote consultation and diagnosis systems for medical applications, contract negotiation and distributed interactive

simulation and much more. Many of these applications disseminate and exchange sensitive and/or classified information. Therefore, basic security services — such as traffic integrity, entity authentication, and confidentiality — are necessary for these collaborative applications.

These security services can be facilitated if the authorized group members share a common secret. This secret key should be updated according to the current *view* of the *session*[1] and is commonly referred to as a *session key*.

The goal of using session (short-term) keys instead of using long-term keys directly is fourfold:

- To limit the amount of cryptographic material encrypted with the same key
- To limit the exposure due to loss of the key
- To avoid long-term storage of secret keys
- To create independence between different and unrelated sessions

The establishment of such temporary keys often involves interactive cryptographic protocols. These protocols should ensure that all the required security properties, such as the authenticity and freshness of the resulting session key, are guaranteed. Such protocols are referred to as *key establishment protocols* which are the focus of this thesis.

Definition 2.1 *[52] A key establishment protocol is a process whereby a shared secret key becomes available to two or more authorized parties for subsequent cryptographic use.*

[1]The determination of the view of the current session is a very important issue, especially in the case of dynamic group settings. This issue will be explained later.

Key establishment protocols can be subdivided broadly into two sets of protocols: key transport (centralized key management) and key agreement (contributory key management):

Definition 2.2 *[52] Key transport protocol is a key establishment protocol in which one party, usually referred to as the key generation center (KGC), creates or otherwise obtains a secret value, and securely transfers it to the other(s).*

Definition 2.3 *[52] Key agreement protocol is a key establishment protocol in which two (or more) parties contribute information, which jointly establishes the shared secret key, (ideally) such that no party can predetermine the resulting value.*

Most of the proposals for group communication security, that have been published so far, have focused on centralized key management where private-key cryptography has been exploited. Mainly the reason for that choice is due to the fact that typically these protocols are less expensive in resources (bandwidth, computation requirements) than their key agreement counterparts. While the centralized approach works reasonably well for static groups and/or one-to-many communication settings, it turns out that contributory key agreement protocols are superior for collaborative applications that are characterized by a dynamically changing membership. The reasons for this are as follows:

- Centralization violates the peer nature of the group by concentrating all the key generation and authentication in a single point
- A centralized key server becomes both a single point of failure or performance bottleneck and an attractive attack target.[2]This is especially the case

[2]It is evident that a key server can be efficiently replicated to address these issues, but it is very costly and difficult (if not impossible) to guarantee the key server availability in any and all partitions of a network.

when the group size is large and if this server serves as the key generation/distribution center for multiple groups.

- KGC needs to establish a pairwise secure channel (by sharing a secure long-term key) with every group member in order to securely distribute keys. This can become prohibitively expensive and an impractical assumption. Moreover, as explained later, this long-term key can be problematic in both the security of the protocol and storage requirements for users

- Some group communication environments, such as ad hoc wireless networks, are highly dynamic and no group member can be assumed to be available at all times.

- There is no way to guarantee the freshness and randomness of the generated key[3], which may be important in some applications (e.g., in the case where the key is later used for a digital signature).

- Some applications require mutual authentication between each pair of members in the group, which is very difficult to satisfy using centralized key distribution due to the fact that authentication will be achieved only through the central authority.

From the security point of view the aforementioned reasons affect the security of group communication services. However, it should be emphasized that there are situations where despite these requirements, the use of KGC may be advantageous. For example, there may be cases where the abilities of protocol participants are not as comparable to those of a KGC in generating a good random number. This problem shows up in some cryptosystems where the selection of good (random,

[3]The user accepts the keying material from KGC without being able to check its freshness and/or randomness

fresh, robust) keys involves more than simply generating random numbers. Factors such as these should be taken into account when choosing between key agreement protocols and the use of KGC. Nevertheless, this situation still can be dealt with using our proposed framework, for example the HGKM protocol can deal with this situation (see Chapter 3).

2.2 Authentication and Key Agreement Protocols

Generally speaking, authentication is central to security. Involvement of many parties in a group communications setting may create a potential for malicious behavior by a dishonest adversary. Authentication can deal with different cryptographic primitives such as access control, authentication of entities, data origin or keys to non-repudiation. The focus of this thesis is limited to authentication of (session) keys or, more precisely, (authenticated) key establishment. To avoid any confusion between entity authentication and authenticated key establishment, we begin with the following definition:

Definition 2.4 *[52] A protocol providing entity authentication informally means that a party successfully engaging another party in such a protocol can be assured (through acquisition of corroborative evidence) of the other party's identity involved in the protocol, and its active presence during the protocol.*

Considering potential applications of such a mechanism, it is clear that entity authentication cannot be seen in isolation and must be considered in a wider context. Mostly, entity authentication is required as part of a session of subsequent and separate actions over some form of channels and the authenticity has to be extended over the complete lifetime of the session. In cases where physical properties of the underlying channel (e.g., separate and tamper-resistant wires used exclusively

to connect two secure access points) guarantee the integrity of a channel and its unique assignment to a particular session, an entity authentication protocol might be sufficient to ensure the authenticity over the lifetime of a session. This is a common assumption in most of the previous key agreement protocols, in spite of the difficulty of establishing such kinds of channels, especially in an open network. The drawback to such an assumption is that one has to be very careful to make sure that apparent end-points of the channel really correspond to the authenticated parties. Otherwise, there is a considerable danger that one falls prey to a man-in-the-middle attack (which is a common attack to most key agreement protocols) where the adversary transparently passes the protocol messages of the entity authentication protocol affecting subsequent actions.

From the above discussion, it is easy to see that authentication has to be securely tied to the session keys, that is, an authenticated key establishment protocol is required. In other words, in designing a key agreement protocol, we have to address two related issues: authentication and key secrecy. However, authors of some protocols (like the Diffie-Hellman protocol [28] and its extensions) presume the existence of authentic channels and consequently claim that their protocols are secure against active attacks using this assumption.

2.3 Two-Party Key Agreement Protocols

Before discussing the characteristics of key agreement in group settings, we first introduce the necessary terminology and give intuitive and informal definitions of the different properties and requirements needed for the classical two-party key agreement setting.

Generally, the main goal of a key agreement protocol can be simply stated as follows: two parties (who have not met before) initiate a procedure to collabora-

tively generate a secret material (which will be used to generate a common session key) over an insecure network [13]. While this goal is clear, less clear are the security requirements that the protocol should satisfy to avoid many possible attacks. A key agreement protocol should provide specific security properties to achieve its main goal. First of all, the resulting session key has to be new and independent of other session keys. This property is usually called *key freshness*. Furthermore, the session key should be known only to the involved parties. This aspect of *key secrecy* is often phrased as the inability of adversaries to learn the (complete) key. However, this formulation has its problems as it presupposes that the leakage of partial information (this would not be ruled out by such a definition) on the session key has no effect. For example, consider an application which naively splits a session key in half into an encryption key and a key for a message authentication code. The aforementioned secrecy definition ensures that an attacker is not able to get both keys. However, it does not prevent the attacker from learning either one of them. This clearly defeats the security of such a system. Similarly, the direct use of the session key in a key establishment protocol message (as often done in the past, e.g., for key-confirmation) might violate the security of a higher-level protocol relying on the resulting session key. Therefore, every single bit of the resulting session key should be unpredictable, a formulation in which the usual complexity-theoretic setting can be traced back to the semantic security (a slightly different formulation of an equivalent meaning and a more commonly used term) introduced by Goldwasser and Micali in [30].

We should emphasize that the modern key agreement protocols should consider the existence of an active attacker who usually has access to the network and who may be able to, in addition to the aforementioned capabilities, steal information, e.g., session or long-term keys, from honest parties and may even corrupt parties and learn their current state including their keys. Clearly, any key agreement proto-

col (actually any cryptographic protocol) cannot provide a security guarantee for a session where the session key is stolen or for a party once is corrupted. However, depending on the impact of the loss of keys on other and past sessions, we can classify key agreement protocols as follows. If the compromise of long-term keys cannot result in a compromise of past session keys of a particular protocol, we can say that this protocol offers *perfect forward secrecy* (PFS) [31]. Conversely, if the compromise of session keys of a particular protocol allows (1) a passive adversary to compromise keys of other sessions, or (2) an active adversary to impersonate the identity of one of the authorized protocol parties in other sessions, we say that this protocol is vulnerable to a *known-key attack* (KKA) [19, 75].

Implicitly, the term "key secrecy" already includes some notion of authentication. We require that only the intended principal be able to learn the key (or any information thereof). This is called *implicit key authentication*. If, in addition, the protocol confirms the active and current participation of the principal in a particular session, it will be referred to as *explicit key authentication*. This can be seen as a special form of entity authentication which additionally provides the establishment of a secure session key. Usually, this is achieved with a *key confirmation*, a protocol which shows evidence that the (same) session key is also possessed by the principal. While we usually cannot enforce *liveness*, (i.e., a guarantee of a successful and timely termination of the protocol) and key confirmation does not prevent a principal from crashing immediately afterwards, it can still form a useful basis in implementing robust and fail-safe applications. For example, honest parties might always write the application context, including the session key, to stable storage before sending a key-confirmation and make all efforts to recover such sessions after a crash; insofar, the key-confirmation would signal the successful and reliable establishment of the related session. Finally, we have to consider the reciprocity of the authentication. Usually, both parties want to authenticate each other including

the common session context, e.g., they need an agreement on all information (explicitly or implicitly) visible at the service interface such as the particular session, both of their identities and the common key. This is called *mutual key authentication.* If authentication is one-sided, such as is typically the case for SSL-based web applications where only server authentication is used, we talk about *unilateral key authentication.* Finally, we have to mention that these security requirements should be achieved in a very adverse environment. In conclusion, the judgment of any protocol, whether it achieves its security requirements or not, should be based on the theory of the protocol and not on its implementation.

2.3.1 Diffie-Hellman Key Exchange Protocol

The first practical solution to the key agreement problem was described by Diffie and Hellman in their seminal paper [28] published in 1976. This solution allows two parties, never having met before nor having shared any secret keying material, to agree upon a shared secret by exchanging a certain number of messages over an open channel. The basic version of the protocol, depicting two stations A and B, is described in Algorithm 2.1.

Algorithm 2.1 *Diffie-Hellman Key Exchange Protocol*

INPUT: A large prime number p *and a generator* α *of* $\mathbb{Z}_p^*$

OUTPUT: $S_{AB} = S_{BA} = \alpha^{r_B r_A} \ mod \ p$,

1. A selects $r_A \in \mathbb{Z}_p^*$

2. $A \rightarrow B : P_A = \alpha^{r_A} \ mod \ p$

3. B selects $r_B \in \mathbb{Z}_p^*$

4. $B \rightarrow A : P_B = \alpha^{r_B} \ mod \ p$

5. A computes $S_{AB}(A) = P_B^{r_A} = \alpha^{r_B r_A} \ mod \ p$,

6. B computes $S_{BA}(B) = P_A^{r_B} = \alpha^{r_A r_B} \ mod \ p$

We observe that $S_{BA} = S_{AB}$ because multiplication is commutative. This enables A and B to communicate with encrypted messages using any private-key cryptosystem with keys S_{BA} and S_{AB}. The security of the protocol is based on the assumption that computing $S_{AB} = \alpha^{r_B r_A} \ mod \ p$ from the intercepted values is as hard as obtaining x from α^x, which is known as the Discrete Logarithm Problem (DLP). So DH protocol is secure against passive attacks, but it does not provide entity authentication or implicit key authentication. DH protocol can easily be compromised by using the man-in-the-middle attack. As a result, DH protocol may be useful in environments where the authenticity of the key tokens (P_A, P_B) are verified using other means. The mathematical background behind this problem is described in detail in Appendix A.

Variants of Diffie-Hellman Protocol

Since the publication of DH protocol, many variants have been proposed to the protocol to reduce its complexity and/or to provide authenticity of the entities carrying out the protocol and/or an implicit authentication of the generated key.

One such variant was proposed by T. ELGamal in [?]. This variant runs in one pass and provides unilateral key authentication but no entity authentication. The protocol assumes that each participant M_i has a long-term key pair $(s_i, P_i = \alpha^{s_i})$, where s_i and P_i are the private and public long-term key respectively. Moreover, each participant has access to authenticated copies of the public keys of the other participants. ELGamal protocol can be slightly modified using digital signatures to provide mutual authentication in one pass. This modification is presented by Nyberg and Rueppel in [55].

In [47], Matsumoto *et al.* proposed several variants to DH protocol. All the variants provide mutual implicit authentication without applying signatures. Algorithm 2.2 explains MTI/A0 protocol and a summary of MTI/A0 and another three related protocols are described in Table 2.1.

Algorithm 2.2 *MTI/A0 Key Agreement Protocol*

INPUT: A large prime p, a generator α *of* $\mathbb{Z}_p^*$ *and an authentic copy of* $P_i = \alpha^{s_i}$

OUTPUT: $S_{AB} = S_{BA} = \alpha^{s_B r_A + r_B s_A} \bmod p$,

1. A selects $r_A \in \mathbb{Z}_p^*$

2. $A \rightarrow B : P_A = \alpha^{r_A} \bmod p$

3. B selects $r_B \in \mathbb{Z}_p^*$

4. $B \rightarrow A : P_B = \alpha^{r_B} \bmod p$

5. A computes $S_{AB}(A) = P_B^{s_A}(\alpha^{s_B})^{r_A} \bmod p = \alpha^{r_B s_A + r_A s_B} \bmod p$

6. B computes $S_{BA}(B) = P_A^{s_B}(\alpha^{s_A})^{r_B} \bmod p = \alpha^{r_A s_B + r_B s_A} \bmod p$,

---

The four schemes proposed by Matsumoto *et al.* are two-pass key agreement protocols and provide mutual key authentication but neither provides key confirmation nor entity authentication.

Recently, Law *et al.* proposed a new authenticated key agreement protocol

Protocol	P_A	P_B	$S_{AB}(A)$	$S_{BA}(B)$	S_n
MTI/A0	α^{r_A}	α^{r_B}	$P_B^{s_A}(\alpha^{s_B})^{r_A}$	$P_A^{s_B}(\alpha^{s_A})^{r_B}$	$\alpha^{r_B s_A + r_A s_B}$
MTI/B0	$(\alpha^{s_B})^{r_A}$	$(\alpha^{s_A})^{r_B}$	$(P_B)^{s_A^{-1}}\alpha^{r_A}$	$(P_A)^{s_B^{-1}}\alpha^{r_B}$	$\alpha^{r_A + r_B}$
MTI/C0	$(\alpha^{s_B})^{r_A}$	$(\alpha^{s_A})^{r_B}$	$(P_B)^{s_A^{-1} r_A}$	$(P_A)^{s_B^{-1} r_B}$	$\alpha^{r_A r_B}$
MTI/C1	$(\alpha^{s_B})^{r_A s_A}$	$(\alpha^{s_A})^{r_B s_B}$	$(P_B)^{r_A}$	$(P_A)^{r_B}$	$\alpha^{s_A s_B r_A r_B}$

Table 2.1: MTI Key Agreement Protocols

in [45], which is known in the literature as MQV protocol[4]. MQV protocol provides key authentication and forward secrecy through a combination of long-term key and ephemeral key pairs[5]. Moreover, the protocol is computationally efficient, bandwidth efficient and involves only group operations and a simple mapping. The only discovered weakness of the protocol is that it can be subjected to *Unknown key-share attack*. This attack is pointed out by Kalisiki in [39]. In a subsequent publication [45], the authors claim that this attack is not a serious attack in practice and can be easily avoided by adding an appropriate key confirmation mechanism to the protocol. The protocol can be defined over any finite commutative group with the hard discrete logarithm problem (multiplicative group or an elliptic curve group, for instance).

Domain parameters of MQV protocol include:

- Q, a base point with large prime order, the generator of $\mathbb{G}$.
- x, the order of the point Q.
- h, the cofactor, defined as $h = |\mathbb{G}|/x$, where $|\mathbb{G}|$ is the order of the group $\mathbb{G}$.

[4]MQV protocol was first published in [51] by Menezes *et al.* and subsequently revised in [44].

[5]Note that the idea of combination of both long-term key and ephemeral key was used before in MTI family but with a different reasoning.

- μ, a mapping from group elements to integers in the range $[2^m, 2^{m+1} - 1]$, where m is the bit-length of $\sqrt{n}$.

MQV protocol is depicted in Algorithm 2.3 defined over the group of points on an elliptic curve.

Algorithm 2.3 *MQV Authenticated Key Agreement Protocol*

INPUT: Domain Parameters

OUTPUT: $S_{AB}(A) = S_{BA}(B) = h(r_A r_B + r_A s_B \mu_B + r_B s_A \mu_A + s_A s_B \mu_A \mu_B)Q$

1. $A \rightarrow B : R_A = r_A Q$

2. $B \rightarrow A : R_B = r_B Q$

3. A checks that $R_B \in \mathbb{G}$ *and* $R_B \neq \mathcal{O}$

4. If the check passes, A computes $S_n(A) = hw_A W_B$

5 B checks that $R_A \in \mathbb{G}$ *and* $R_A \neq \mathcal{O}$

6. If the check passes, B computes $S_n(BA) = hw_B W_A$

Where $w_i = (r_i + \mu_i s_i)$ *mod x and* $W_i = R_i + \mu_i(s_i Q); i = A, B$

2.4 Group Key Agreement

In this section, we will discuss group key agreement protocols, where the emphasis will be on the different nature and related issues of group key agreement protocols to that of a two-party protocol. The main goal, as well as the security properties, for group key agreement protocols are roughly the same as that for the two-party case. However, due to the dissimilarity of characteristics between two-party and n-party settings, some additional precautions should be taken into account when designing group security protocols. Those related to group key agreement protocols are discussed in the following subsection.

2.4.1 Group Communications Security Challenges

The nature of group communications makes them inherently prone to different types of attacks which have no counterpart in unicast and cannot be effectively dealt with by using trivial extensions of techniques used to secure unicast communications. Group communication characteristics and their related attacks can be listed as follows:

- Group communications typically involves a large number of principals and this can potentially make it easier for an attacker to pose as another legitimate principal or try mount parallel attacks. In fact, in contrast to unicast communications where the communicating parties know the identity of each other, knowledge of the session membership is typically not guaranteed in group communication, and session membership is hard to be controlled.

- Group communication data are transmitted over many channels of the network at the same time, which presents many more opportunities for interception of traffic.

- There is no mechanism that can prevent either a legitimate group member, or non-group member, from overwhelming the group members with spurious data, which can result in denial of service attacks.

- When an attack does take place, the group communication setting can easily result in a large number of group members being affected adversely.

Besides the aforementioned dissimilarities, there are other differences which have to be considered. On one hand, due to the dynamic nature of group composition, it is more difficult to guarantee the honesty of parties. A party might be considered trusted in respect to its "legal" membership period. However, it also

might try to misuse this time to attempt to gain access to the group at other "illegal" membership periods, potentially even in coalition with other former group members (these types of attacks will be formally discussed in Chapter 5). In fact, a number of group key establishment protocols actually fall prey to such coalition attacks (see for example [18, 22, 71]). On the other hand, the notion of key authentication has to be broadened. In the two-party case it is natural to always require that both parties agree on each others identities, at least for mutual key authentication[6]. This is very hard to achieve in the n-party case. Nonetheless, one can also imagine a weaker form of key authentication where group members are just assured that only legitimate group members are present without necessarily getting any knowledge of the actual group membership of a session. The former will be called *mutual group key authentication* whereas the latter will be simply called *group key authentication*. Group key authentication is sufficient and it is the only practical form of authentication in the case of large asymmetric groups where a static party controls access to the group and members do not know each other, e.g., in video-on-demand applications. However, in dynamic groups, where the roles of group members are symmetric and a common agreement on the group membership is essential, mutual group key authentication is more desirable and more natural than simple group key authentication. In groups, the verification of the authenticity does not always have to be direct as in the two-party case. It also can be indirect via some other group member(s)[7]. This requires additional trust assumptions from these intermediary group members. Nonetheless, these trust assumptions are quite natural as we do already trust insiders not to give away the common group key. Similarly, in extending explicit key authentication we have the option of requiring a confirmation which is

[6]Unilateral key authentication does not seem to have an equivalent in a group setting.

[7]For example, if principal A authenticates principal B and B authenticates C then A authenticates C and so on.

either direct and pairwise or indirect e.g., over a spanning tree on the membership graph.

2.4.2 Fault-Tolerance

Several group key agreement schemes have been proposed in the literature [37, 64, 19, 38, 65, 9]. However, none have been practically deployed. One of the main reasons (besides inefficiency and proneness to attacks) is the use of a logical flat setting, wherein a link or member failure will affect the whole protocol. In practice, group key establishment is typically done in a centralized manner [33, 74, 72]: one dedicated party (typically, a group leader) chooses the group key and distributes it to all group members. This is actually key transport (often also called key distribution in such a context), not key agreement. As previously mentioned, a permanently fixed group leader is a potential performance bottleneck and a single point of failure. Some group communication environments (such as ad hoc wireless networks) are highly dynamic and no group member can be assumed to be present all the time. This is also the case in wired networks when high availability is required. Therefore, one of the better approaches to fault-tolerance (such as handling network partitions and other events) is to adopt a clustering structure in the proposed key agreement protocols. Using clustering, one is able to easily isolate any member or link failure away from the other group members. Secure group key agreement protocols, such as HGKM and CGKA (presented later in Chapter 3) are fault-tolerant in terms of integrity and confidentiality, e.g., the authenticity and secrecy of the key is ensured. In addition using a reliable group communication system can improve the availability or liveness of a given node, prevent denial of service attacks, and facilitate administration.

2.4.3 Management of Groups

There is no inherent reason to require a single group leader to make the decisions as to whom to add to, or exclude from, a group. Ideally, decisions regarding group admission control, e.g., who can be added to or removed from a group and who can coordinate such operations, should be taken according to some local group policy (see, for example, Kim *et al.* [42] for a good discussion on this issue) and should be orthogonal to the actual key establishment protocol deployed.

Although we argue in favour of distributed key agreement, we also recognize the need for a central point of control (it will be referred to as network administrator (*NA*) in the rest of the thesis) for group membership operations such as adding and excluding members[8]. This type of a role (Network Administration) serves only to coordinate and synchronize the membership operations and prevent chaos. However, the existence and assignment of this role is orthogonal to key establishment, i.e., it should not affect the security and performance of the key establishment protocol. Moreover, the responsibility of this role can be changed at any time and it is largely a matter of policy.

2.4.4 Handling Membership Changes

As previously mentioned, the membership of the group is mutable, i.e., during the course of the group, a new authorized member may join in or an old member may leave (or get evicted from) the group. Also, due to possible network failures, the whole group can be split into two (or more) smaller subgroups. After the network heals, the subgroups may merge again. These membership changes have to be re-

[8]In practice, such a server, that works as *NA*, can be distributed or replicated where each *NA* is responsible for its nearest cluster(s). In this case the other *NAs* should coordinate among themselves to ensure consistency and uniqueness of multicast addresses.

flected in a set of corresponding auxiliary protocols as well. After any membership changes, the session keys should be changed accordingly. In the following, we distinguish between *initial key agreement* (IKA), the main key agreement protocol, which is conducted at a group genesis, and *auxiliary key agreement protocols* (AKAs). AKAs encompass all operations that regenerate the group session key after group membership changes such as member addition and exclusion or when key refresh is required.

Initial Key Agreement (IKA) Protocol

IKA takes place at the time of group genesis. On the one hand, this is the time when protocol overhead should be minimized for rapid and efficient group creation. On the other hand, for highly dynamic groups, certain allowances should be made. For example, extra IKA overhead can be tolerated in exchange for lower AKA (subsequent key agreement operations) costs. However, one should note that it is the security of the IKA, not its overhead costs, that should be the overriding concern. Naturally, IKA requires contacting every prospective group member to obtain a key share from each member. Hence, it may be possible to coincide (or interleave) the IKA with other security services such as access control.

Auxiliary Key Agreement (AKA) Protocols

As mentioned above, the membership of the group is dynamic in nature, and in order to guarantee the security of the group communication, the session key should be updated after every membership change. The easiest way to do that is to perform the IKA after every membership change, a trend that may affect the system performance. Other auxiliary protocols can also help in the updating of the required key. In this section, a discussion of auxiliary group key agreement protocols and

the related security issues will be presented.

The crucial security requirement to all AKA protocols is key independence. Informally, it encompasses the following two requirements, which are closely related to PFS and in particular to that of KKA:

- Old group keys used in past sessions must not be discovered by new group member(s). In other words, a group member must not have knowledge of (or be able to calculate) keys used before it joins the group.

- New keys must remain out of reach of former group members, i.e., members excluded in past sessions should not have access to new keys.

While the requirement of key independence is fairly intuitive, we need to keep in mind that, in practice, it may be undesirable under certain circumstances. For example, a group conference can commence despite the fact that some of the intended participants may join in at a later time. Upon their arrival, it might be best not to change the current group key to allow the tardy participant(s) to catch up. In any case, this decision should be determined by a policy local to a particular group.

The available membership changes include: member addition and member exclusion. The former is a seemingly simple procedure of admitting a new member to an existing group. The main security requirement of member addition is that the new member should not have any access to the previous session keys. Member exclusion is also relatively simple and it takes place either voluntarily or by force (a member may be evicted because of many reasons). In either case, the security implications of member exclusion are the same, i.e., an excluded member should not have access to any future key unless it is re-admitted to the group.

Membership changes can take place in bulk, namely, bulk addition and bulk exclusion. Bulk addition, in turn, has multiple variants:

- Bulk join: the case of multiple new members who have to be brought into an existing group and, moreover, these new members do not already form a group of their own. Also we can consider an independent multi-member addition within a certain short period as bulk join.

- Group merge: the case of two groups merging to form a super-group; perhaps only temporarily. Or it may take place after a network failure, which resulted in a group partition (see below), heals.

Similarly, bulk exclusion can also be thought of as having multiple flavors:

- Bulk leave: multiple members may be excluded at the same time. To exemplify, consider a teleconference meeting between managers of a virtual corporation who need some outside experts opinions, but do not want these experts to learn about other topics they are discussing. Readmitting these experts again can be considered a bulk join (see above).

- Group partition: a monolithic group needs to be broken up in smaller groups, which may be the result of a network failure or a redistribution policy.

Although the actual protocols for handling all bulk changes may differ from those of single member, the salient security requirements (key independence) remain the same.

Besides the above mentioned membership changes, it is often necessary to perform a routine key change operation. This may be necessary due to, for example, a local policy that restricts the usage of a single key by time or by the amount of data encrypted or signed using this key. To distinguish this from key changes due to membership changes, this operation will be referred to *key refresh* in the rest of the thesis.

2.4.5 Complexity Measures and Metrics

Generally speaking, group key agreement protocols are multi-round protocols. Consequently, protocol complexity is of great concern due to the direct relationship of the number of participants to the computation and communication complexity. In the previous section, we described the required group key agreement services and desirable properties. Clearly, we have to consider the cost and performance of protocols to estimate their practicality, their scalability[9] and their suitability to particular environments. In the following discussion, the round can be defined as the protocol action which can not be processed unless its previous action has been successfully completed. In key agreement protocols, a new round is initiated each time any number of members need to provide additional pieces of information to help other members in their key computations.

Generally, key agreement protocols consist of two main stages:

- Contributions collection: In this stage all group members contribute their secret share to the other members in such a way that it enables the other members to calculate the common secret. Generally, this stage can be done in two different ways. The first one (unicast) is as simple as a member updating the received message from its previous member by adding its secret share and forwarding this piece of information to its neighbour. We can see in this way that the number of rounds will increase as the number of group members increases. In other words, the number of rounds of this approach is related to the number of participants. In the second way (broadcasting), group members (or a subset of the group members) simultaneously broadcast their contributions to the other group members. These contributions may be

[9]Generally speaking, the scalability can be considered as the ability of the protocol to perform well with large size input.

based only on the members secret shares or on their current information.

- Common secret calculation: Upon receiving the other group members' contributions, each member (using its secret share) calculates its view $S_n(M_i)$ of the common secret. We can say that the protocol is successfully completed if the views of the common secret of all group members are equal and no one outside the group can get access to this common secret.

From the conceptual prospective, we are interested in two major cost aspects: the computation and communication costs. There is always a trade off between both costs, but this is typically based on the underlying communication facilities and on the applications. For example, the cost of communication in a modern high-speed LAN setting can appear negligible in comparison to the cost of modular exponentiations. On the other hand the communication cost is very important in high-delay networks (e.g., WANs, the Internet). The transmission delay always has an upper bound related to the speed of the signal through the communication medium.

Typically, it is impossible to estimate concrete computational costs as many cost-critical aspects are implementation-dependent and some primitives are difficult to compare. However, we should get a good base for comparison by identifying the expensive and time-critical operations, and consider them individually. Furthermore, computational capabilities might largely differ among the involved parties. However, in most group communication applications (like audio/video conferences for instance), all group members are equal and we can safely assume that they have similar computational capabilities[10]. Therefore, we will focus on complexity measures of both computation load on individual group members in terms of the number of expensive operations, and the lower bounds on the duration of a protocol where

[10]Although, the proposed approach can be modified to support a heterogenous system with different computing capabilities like PDAs.

we only consider the sum of serial operations. Regarding the computational cost, we will consider the cost of modular exponentiations, since it is the most costly computational process in most public-key based protocols. With respect to communication costs, the impact of a protocol clearly depends on the topology and properties of the network and the group communication system being used. Primarily the critical aspects are latency and bandwidth. Unfortunately, they cannot be measured directly. The approach used here will be to list for each protocol the number of messages, their cumulative size, and the number of rounds. Additionally, we will distinguish between unicast and multicast messages, i.e., messages from one member to another and from one group member to the rest of the group, respectively. From these numbers we can then derive an estimate on the concrete protocol latency and network load when the actual networking environment is known.

In the ensuing analysis, the following costs will be considered:

Number of Rounds (R)**:** The number of rounds is obviously a measure of the time needed to carry out the protocol ([54] p.133–134). In other words, the total number of rounds is the number of serial actions among members.

Number of Messages (M)**:** The total number of messages and the total number of broadcasts sent according to the protocol. A message is a single packet sent from one member to another, while a broadcast is a message sent by a member and received by a subset (or all) of the group members. So the number of messages measures the number of packets sent.

Number of Exponentiations (Ex_S)**:** Since Modular Exponentiation is the most expensive operation (it requires $O(log^3 p)$ bit operations in $\mathbb{Z}_p^*$), the major concern in reducing the computation overhead is to reduce the total number of exponentiations needed to be executed by members during the protocol.

Maximum Message Size (M_s): The maximum message size is considered as the size of the largest message sent during the protocol processing. From the largest message size we can calculate the cumulative message size. From this we can easily derive the concrete bandwidth requirement once the concrete parameters, such as the group $\mathbb{G}$ and its encoding, are known.

2.5 Related Work

Since the publication of 2-party Diffie-Hellman key exchange (DH) in 1976, various solutions have been proposed to extend this mechanism to an n-party Key Agreement Protocol. The earliest attempt to provide contributory group key agreement is due to Ingemarsson *et al.* [37]. This protocol requires synchronous startup and executes in $(n-1)$ rounds. Members must be arranged in a logical ring. In a given round, every participant raises the previously received intermediate key value to the power of its exponent and forwards the result to the next member. After $(n-1)$ rounds, every member has computed the same key S_n. This work did not address the case of rekeying after membership changes.

Another interesting scheme was presented by Steer et al. in [64]. The protocol (referred to as STR) takes n rounds to complete. In some ways STR is similar to GDH.2 [5], which will be discussed later. Both take the same number of rounds. Also, both accumulate keying material by traversing group members once per round. However, the generated session key in STR has a very different structure, given by: $S_n(M_i) = \alpha^{r_n\alpha^{r_{n-1}\alpha\ldots r_3\alpha^{r_1 r_2}}}$. Interestingly, STR is well suited for adding new members, it takes only two rounds to add a new member. Member deletion, on the other hand, is difficult in STR since there is no natural group controller.

Another notable result is due to Burmester and Desmedt (BD)[19] whose protocol executes in only three rounds:

1. Each M_i generates its random exponent r_i and broadcasts $Z_i = \alpha^{r_i}$.

2. Each M_i computes and broadcasts:

$$X_i = (\frac{Z_{i+1}}{Z_{i-1}})^{r_i}$$

3. Each M_i can now compute the key:

$$S_n(M_i) = (Z_{i-1})^{nr_i}.X_i^{n-1}.X_{i+1}^{n-2}\ldots X_{i-2} \; mod \, p$$

The key defined by BD is different from all protocols discussed thus far, namely:

$$S_n = \alpha^{r_1r_2+r_2r_3+\ldots+r_nr_1}.$$

The protocol has been proven secure given that the Computational Diffie-Hellman (CDH) problem is intractable. The CDH problem can be simply stated as follows: Given $\alpha^x \; mod \, p$ and $\alpha^y \; mod \, p$ it is hard to find $\alpha^{xy} \; mod \, p$. While the BD protocol is efficient and secure, it is not well suited for dynamic groups.

Steiner *et al.* have addressed the dynamic membership issues in their group key agreement proposal [5] and have proposed a family of Group Diffie-Hellman (GDH) protocols (namely, GDH.1, GDH.2, and GDH.3). Their protocols can be considered as a generic extension of the two-party Diffie-Hellman [67]. GDH provides contributory authenticated key agreement, key independence, key integrity, resistance to known key attacks, and perfect forward secrecy. Their protocol suite is fairly efficient in leave and partition operations, but the merge protocol requires as many rounds as the number of new members to complete the key agreement procedure. One of their basic group key agreement protocols (without authentication) is briefly described in the following.

We will explain the Group Diffie-Hellman protocol GDH.2, which is the most practical protocol of the family, proposed in [5] (see Algorithm 2.4). Let p be

a prime integer and q be a prime divisor of $p-1$. Let $\mathbb{G}$ be the unique cyclic subgroup of $\mathbb{Z}_p^*$ of order q, and let α be a generator of $\mathbb{G}$. The authors assume that M_n shares (or is able to share) with each M_i a distinct secret K_{in}. The protocol consists of two stages, upflow and downflow. The purpose of the upflow stage is to collect contributions from all group members. As shown in Algorithm 2.4, M_i receives the contribution from M_{i-1} and composes i intermediate values (each with $(i-1)$ exponents) and one cardinal value containing i exponents. For example, if M_4 receives the set:

$$\{\alpha^{r_1r_2r_3}, \alpha^{r_1r_2}, \alpha^{r_1r_3}, \alpha^{r_2r_3}\}$$

then it outputs the set:

$$\{\alpha^{r_1r_2r_3r_4}, \alpha^{r_1r_2r_3}, \alpha^{r_1r_2r_4}, \alpha^{r_1r_3r_4}, \alpha^{r_2r_3r_4}\}$$

The cardinal value in this example is $\alpha^{r_1r_2r_3r_4}$. By the time the upflow reaches M_n the cardinal value becomes $\alpha^{r_1\cdots r_{n-1}}$. Thus M_n is the first group member to compute the key $S_n = \alpha^{r_1r_2\cdots r_n}$. At the final step of the upflow stage, M_n computes the last batch of intermediate values. In the second stage M_n broadcasts the intermediate values to all group members. By receiving these values each group member can calculate the session key S_n.

Algorithm 2.4 *Group Diffie-Hellman Protocol GDH.2*

INPUT: A large prime p, a generator α *of* $\mathbb{Z}_p^*$

OUTPUT: $S_n = \alpha^{r_1 r_2 \cdots r_n} = S_n(M_i)$

First stage :(Upflow): round i; $i \in [1, n-1]$

1. M_i *selects* $r_i \in \mathbb{Z}_p^*$
2. M_i: $\{\alpha^{\frac{r_1 \cdots r_i}{r_j}} |_{j \in [1,i]}, \alpha^{r_1, \cdots, r_i}\} \xrightarrow{\qquad} M_{i+1}$

Second stage: (Broadcast): round n

3. M_n *selects* $r_n \in \mathbb{Z}_p^*$
4. M_i $\xleftarrow{\{\alpha^{\frac{r_1 \cdots r_n}{r_i}} |_{i \in [1,n]}\}}$ M_n

Upon receipt of the above, M_i *computes:*

$\alpha^{\{\frac{r_1 \cdots r_n}{r_i}\} r_i} = \alpha^{r_1 r_2 \cdots r_n} = S_n(M_i)$

In the case of a member addition, under the GDH.2 protocol, the last member M_n saves the contents of the upflow messages. M_n generates a new exponent $\widehat{r_n} \in \mathbb{Z}_p^*$ and computes a new upflow message and sends it to the new member, M_{n+1}. Then M_{n+1} generates its own exponent and computes the new key. Finally, under the normal operation of the protocol (where M_{n+1} takes M_n's rule), M_{n+1} computes n intermediate values and broadcasts to all the other group members. Roughly speaking, the communication and computation costs of performing GDH.2 increase linearly with the number of the group members (group size). This will affect the efficiency of the protocol, especially when the group size increases.

The previous proposals assume that the group members are arranged in a logical ring. This setting, which we will call a *flat setting* during the thesis, is not resilient to a link or member failure. In other words, if any link or member has failed either intentionally or accidentally, the whole protocol halts and must be repeated. Moreover, the efficiency of these protocols (except BD protocol [19]) is related to

the group size. To solve these problems, there are many solutions that are based on a distributed or *hierarchal* setting. One of these solutions has been proposed by Becker and Wille [9]. In their paper, the authors systematically analyze the communication complexity of the initial key agreement for contributory group key agreement protocols. They prove the lower bound for various measures and describe a novel protocol, $2y$-octopus as an example, requiring only the minimum number of rounds ($y = \lceil \log_2 n \rceil$). Their main idea is to arrange the parties on a y-dimensional hypercube, i.e., each party is connected to y other parties. The protocol proceeds through y rounds, in the j^{th} round each participant performs a two-party DH with its peer on the j^{th} dimension, using the key of the $(j-1)^{th}$ round as its secret exponent. The exponents of the first round are chosen at random by each party. For example, the resulting key for a group of 8 parties is: $S_8 = \alpha^{(\alpha^{(\alpha^{(r_1 r_2)} \alpha^{(r_3 r_4)})} \alpha^{(\alpha^{(r_5 r_6)} \alpha^{(r_7 r_8)})})}$. While adding new members and in particular groups is easy with $2y$-octopus, it fails completely in terms of member exclusion. Splitting the group on the y^{th} dimension into two halves seems the only efficient exclusion procedure.

Another interesting protocol which addresses the scalability issue in its key agreement protocol TGDH is proposed by Kim *et al.* in [43]. TGDH is an adaptation of key trees in the context of fully distributed, contributory group key agreement. Their group key agreement uses Diffie-Hellman key exchange in a binary tree. The tree is organized in the following manner: each node $\langle l, v \rangle$ is associated with a key $K_{\langle l,v \rangle}$ and the corresponding blinded key $BK_{\langle l,v \rangle} = g^{K_{\langle l,v \rangle}} mod\ p$. The key at the root node is the group key shared by all members, and a key at the leaf node is a random session contributed by a group member. The basic idea is that every member can compute a group key when all blinded keys on the key tree are known. After any group membership event, every member unambiguously adds or removes some nodes related with the event and invalidates all the affected nodes.

Another solution to the scalability problem, Iolus, was proposed by S. Mittra

in [53]. In Iolus, the group members are partitioned into clusters where each cluster is assigned a cluster controller. Each cluster has its own cluster session encrypting key and a cluster key encrypting key. The cluster controller generates the cluster keys, and securely distributes them to its cluster members, and hence it can be considered as a distributed transfer protocol not a key agreement protocol. Moreover, intra-cluster communication should be controlled through clusters controllers since there is no global group session key known by all group members.

2.6 Summary

In this chapter, the dimensions of key agreement protocol has been presented, where a discussion of the requirements and the main goal of key agreement protocols has been presented. Then, detailed discussion of the many different issues related to group key agreement protocols are also discussed. To demonstrate the different approaches in this field, some proposals have been briefly described. Although there are many other proposals in the literature, they follow similar (or slight modified) approaches as previously described. There are many common drawbacks in these approaches, which will be tackled through the rest of the thesis. These drawbacks can be summarized as follows:

- Efficiency: As described, the complexity of previous protocols is related to the group size. As a result most of these protocols do not scale well for a large group size.

- Applicability: BD protocol assumes simultaneous broadcasting, which is not applicable (or desirable) in all applications. Moreover, assuming distribution of the group members in a fixed topology is not desirable in some applications.

- Robustness: Due to the group members being distributed in a flat setting, a link failure or compromised member affects the whole group and invalidate the group key.

- Security: Most of these protocols presume that their security can be easily reduced to the two-party key exchange, which is not the case as will be shown in Chapter 5.

Chapter 3

Efficient Group Key Agreement Protocols

In this chapter, a novel framework for scalable key management protocols in group communications is presented. The proposed framework is based on the distribution of members of the communicating group into nested clusters. Two different key agreement protocols (HGKM, CGKA), based on members clustering scheme, are described. The auxiliary protocols needed, when the membership changes, are also given. From the complexity analysis of the proposed framework, it will be shown that it is scalable to large groups with frequent membership changes.

3.1 Introduction

As explained in the previous chapter, there is a common deficiency in the existing key agreement protocols : they do not scale well for large size groups. In this chapter a new approach, which aims to improve the scalability of key agreement protocols, is presented. We will address the question of scalability in key agreement

protocols for group communication, that is the efficient generation/regeneration of a group session key within a large group size and/or a group with frequent membership changes.

The main idea behind this approach is based on a multi-level distribution of the universal group into small, size-bounded subgroups or clusters. The motivation for this approach is that the key generation/regeneration cost (communication and computation overhead), of the most efficient key agreement protocols that have been published, is proportional to the group size n. Using the hierarchical system, the cost of generating/regenerating the group session key increases logarithmically (or less) with the group size. This improvement comes from using concurrent processing for the same level's clusters, which reduces the communication and the computation overhead needed to generate/regenerate the group session key. This feature, from the complexity point of view, reduces the time required to generate the session key and distributes the load of computations among the group members.

3.2 Clustering Techniques

The concept of dividing the geographical region to be covered into small zones (as shown in Figure 3.1) has been presented implicity as *clustering* in the literature [23]. There are two types of clustering techniques: Partitional and Hierarchical. Their definitions are as follows [32]:

Definition 3.1 *A partitional clustering algorithm constructs k partitions out of the total number of n members, where each cluster optimizes a clustering criterion, such as minimization of the transmission delay between members of the cluster. One of the issues facing such algorithms is their high complexity, as some of them exhaustively enumerate all possible groupings and try to find the global optimum.*

Definition 3.2 *Hierarchical clustering algorithms create a hierarchical decomposition of the objects.*

Hierarchical algorithms can be either *agglomerative* (bottom-up) or *divisive* (top-down)

- Agglomerative algorithms start with each member being a separate cluster itself, and successively merge groups according to a distance measure. The clustering may stop when all members are in a single group or at any other point of the algorithm.
- Divisive algorithms follow the opposite strategy. They start with one group of all members and divide the members in disjoint groups at every step until all members fall into separate clusters. This is similar to the approach followed by divide-and-conquer algorithms. Our construction of the hierarchical structure follows this approach.

Obtaining a hierarchical organization of a network is a well-known problem in distributed computing. Hierarchical structures have been used to provide scalable solutions in many large networking systems that have been designed. It has been proven effective in the solution of several problems, such as minimizing the amount of storage for communication information (e.g., routing and multicast tables) thus reducing information update overhead, optimizing the use of network bandwidth, and distributing resources throughout the network. In wireless networks, partitioning the nodes into groups (clusters) is similarly important, as it is crucial for controlling the spacial reuse of the shared channel and for minimizing the amount of data to be exchanged in order to maintain and control information in a mobile environment. For a complete discussion on clustering in wireless networks, see for example [7].

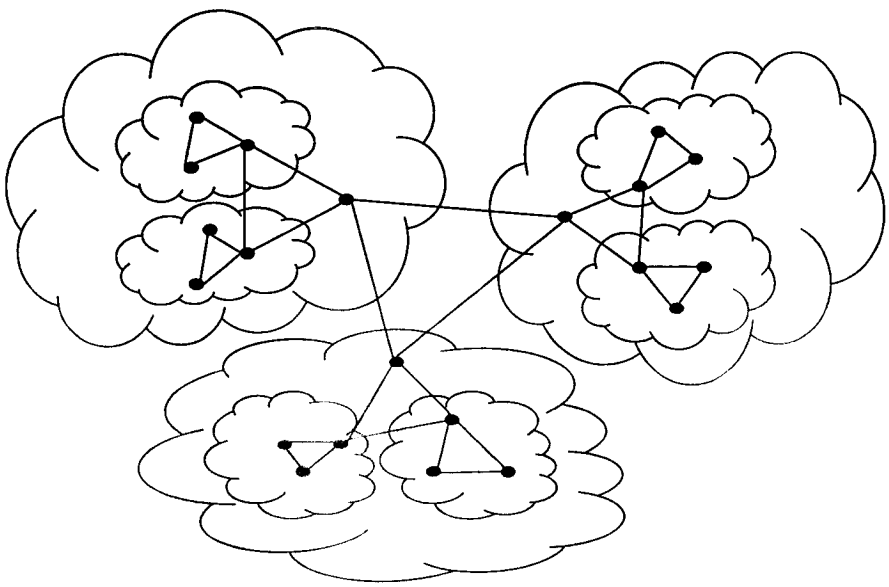

Figure 3.1: Clustering Concepts

3.2.1 Desirable Attributes of the Clustering Scheme

For any clustering scheme there are many desirable attributes that differ according to the application. For the purpose of our key agreement protocol, the clustering scheme should have the following attributes:

- Each cluster is connected. This is an obvious requirement to localize and guarantee inter-cluster traffic.

- All clusters have a minimum and maximum size constraint. A maximum constraint limit the cluster size to within what can preserve the whole protocol efficiency. Ideally we want all clusters to be of the same size so that no cluster is overburdened, nor under-burdened, with processing and storage requirements. Moreover, a large size cluster affects the entire protocol performance.

- Two clusters (in any level) should have no overlap. Meanwhile, two clusters in different levels may have a limited overlap.

These attributes can be stated in a formal way as a generic network clustering problem with certain conditions as follows: Given an undirected connected graph $G = (V, E)$, and a positive integer d, such that $1 \leq d \leq |V|$, where V represents the total number of nodes (vertices) and E represents the connection between every two nodes. Find a collection of subsets (clusters) $C_1, ..., C_{N_C}$ of V, so that the following conditions are met.

1. $\cup_{i=1}^{N_C} C_i = V$ all vertices are part of some cluster.
2. $d \leq |C_i| < 2d$. This is the size bound for the clusters
3. $| C_i \cap C_j | \sim O(1)$. Two clusters should have up to a small number of common vertices (in our clustering algorithm, it is one)

The clustering algorithms differ in the criterion of choosing the suitable members to constitute a cluster and in choosing the cluster leader. These criteria can be one of the following:

- The application: In some applications, the clustering scheme should follow certain rules which are urgent in the fulfillment of the ultimate goal of the application. For example, in some military applications it may be necessary to collect some nodes in one cluster to keep their communication secure even if it will affect the efficiency of the protocol.
- Maximum distance between the neighbours: The transmission delay may be one of the most important criteria in clustering algorithms, so it is often preferred to limit the cluster within a certain geographical area, wherein the maximum distance between cluster members is limited. In wireless applications

this criterion can be applied to number of hops between nodes, where a one hop route is more reliable rather than multi-hop one.

- The unique ID associated to each member: The member with the lowest ID is selected as a cluster leader, then the cluster is formed by that member and all its neighbours.

- The generic weight of group members: The bigger the weight of the member, the more suited the member is for the role of cluster leader. This criterion is applied in clustering algorithms in wireless networks where the weight of the member can be measured as the lifetime of its battery, its computational power or its storage capability.

3.3 Construction of the Hierarchical Structure

The proposed framework assumes distribution of the group members in a hierarchical structure. For this purpose, the structure has to be first constructed. This can be done in several ways. In the initial state, it is considered that the group members are arbitrarily distributed over some region and the group members know their neighbours. It should be mentioned that the definition of a member neighbour is based on the criteria mentioned in the previous section. The resulting hierarchical scheme consists of many levels (as shown in Figure 3.2), each level consists of one or more clusters. Levels are numbered sequentially starting from the top level, $level_1$ and ending at $level_l$. Each member of the group identifies its level. Each member, upon receiving the level announcement message from its parent, obtains its own level by increasing the level of its parent by one, and then announces its own level.

Below, one possible protocol for constructing the required hierarchical structure is described. The clustering algorithm consists of different phases, the first phase

is the Leaders Election Phase. In this phase, the algorithm chooses d members to be the protocol initiators. The distributed algorithm in [24] can be adopted in this phase or it may be easier if these leader are chosen by *NA*. Again, leader election is controlled by the required criteria assigned by the algorithm. The chosen d[1] members constitute the root cluster, $C_{1,1}$. Each member in the root cluster becomes a root (parent) of a $(d-1)$-ary tree and its neighbours become its *children*. To construct our hierarchical distribution, each member in the root cluster sends a message to all its neighbours. After receiving a message, neighbour members acknowledge it. The parent confirms the first $d-1$ messages and the rest are ignored. Upon receiving the confirmation message, these members confirm themselves as a children to that parent. These children construct clusters of $level_2$. The members of $level_2$ clusters send similar messages to all their neighbours (except their parents) to construct $level_3$ clusters. This process will continue until all members of the group participate in the hierarchical structure. The maximum number of levels will be l. For clarity, unless it is stated otherwise, the hierarchical structure used in our framework is arranged to be well balanced, that is, all the clusters have an equal number of members). The hierarchical structure consists of l levels, each level comprising of one or more clusters. Levels are numbered sequentially starting from $level_1$, which consists of one cluster, and ending at $level_l$, which consists of $d(d-1)^{l-2}$ clusters, except the root cluster i.e., $l \geq 2$.

The total number of group members n of our hierarchical structure (Figure 3.2) can be represented as a function of the maximum number of members within the cluster, d, and the total number of levels, l, as follows:

[1]The value of d should be chosen to small to improve the whole protocol performance with minimum value of 3.

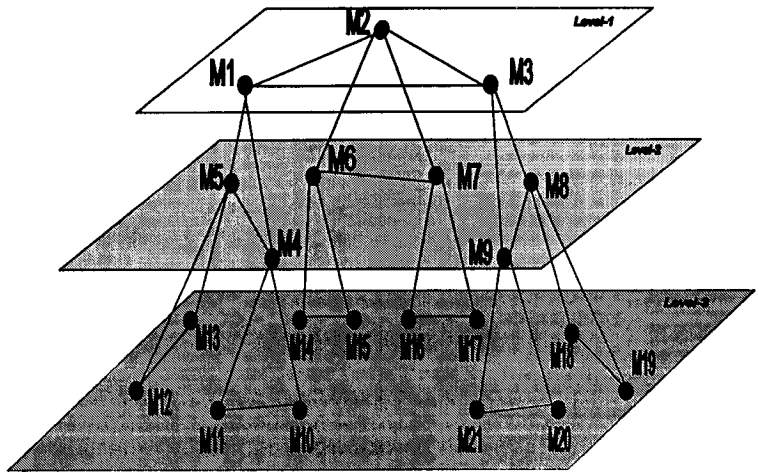

Figure 3.2: Members Distribution

$$n = d\sum_{i=0}^{l-1}(d-1)^i = d\frac{(d-1)^l - 1}{d-2} \tag{3.1}$$

Similarly, the maximum number of clusters, N_C, in the hierarchy can be calculated as a function of (d, l) as follows:

$$N_C = 1 + d\frac{(d-1)^{l-1} - 1}{d-2}. \tag{3.2}$$

From (3.1) and (3.2) we can state the relation between n and N_C as follows:

$$N_C = n - d(d-1)^{l-1} + 1. \tag{3.3}$$

In this hierarchical structure, the indexing scheme of members (clusters) can be used to determine the position of any member (cluster) based on the maximum number of members (d) in each cluster. Moreover, we can identify each member's neighbours in the same cluster. Also from this indexing scheme we can determine the cluster leader chosen by the algorithm, its ancestors up to the root cluster and

also the children's clusters down to the $level_l$. Clearly, there are many indexing schemes that can be used in this hierarchical structure.

After distribution of group members in a multilevel hierarchical structure, the actual key agreement protocol starts. In the following sections, two different possible key agreement protocols, based on clustering of members, are described. In the first protocol, Hybrid Group Key Management protocol (HGKM), the generated cluster session key of the lower level is used by the cluster leader to encrypt the upper level cluster session key. While in the second protocol, Cluster Group Key Agreement (CGKA), the generated cluster session key is mapped to be used by the cluster leader as its secret share to generate the upper level cluster session key. Motivation and possible advantages of each protocol will also be discussed.

3.4 Hybrid Group Key Management (HGKM)

HGKM protocol can be considered as a merge between the two approaches of key establishment, key transport and key agrement. HGKM uses a key agreement protocol to enable the members of each cluster to contributively generate the cluster session key in a certain level. The upper level cluster session key is then transferred to the lower level cluster members by their leader encrypted by their cluster session key. In other words, HGKM is a distributed key transfer protocol, where the responsibility of generating and distributing the global session key is distributed between the clusters leaders instead of one Key Generating Center (KGC). In this protocol, binary indexing is used where a member $M_{i,j}$ is the j^{th} member in the i^{th} level. This indexing scheme determines the position of any member (cluster) in the hierarchy based on the maximum number of members d in each cluster. For example (as shown in Figure 3.3), if we have a hierarchical system with $d = 3$ members in each cluster, $M_{3,12}$ is the 12^{th} member in the 3^{rd} level . Also from this indexing

scheme we can determine the cluster leader and parent clusters up to the root level cluster and also the children clusters down to $level_l$ as was explained before.

We will define two sets of variables which will be used in the following discussion:

- $KeySet(M_{i,j})$: The set of keys that are held by the member $M_{i,j}$. This set can be formally defined as follows:
  $KeySet(M_{i,j}) = \{K_{x,y}$, where $1 \leq x \leq i$ and $y = \lceil \frac{j}{(d-1)^{i-x+1}} \rceil \}$, for example (refer to Figure 3.3) $KeySet(M_{3,8}) = \{K_{3,4}, K_{2,2}, K_{1,1}\}$. Note that each member holds its cluster session key and its upper level (parents) clusters session keys up to the root cluster session key $K_{1,1}$ which is the group session key.

- $UserSet(K_{i,j})$: The set of users who hold the key $K_{i,j}$. Formally it can be identified as follows: $UserSet(K_{i,j}) = \{M_{x,y}$,
  where $i-1 \leq x \leq l$ and $(j-1)(d-1)^{x-i+1}+1 \leq y \leq j(d-1)^{x-i+1} \}$,
  note that $K_{1,1}$, as a group session key, should be held by all the members of the group. Also referring to Figure 3.3, $UserSet(K_{2,2}) = \{M_{1,2}, M_{2,3}, M_{2,4}, M_{3,5}, M_{3,6}, M_{3,7}, M_{3,8}\}$. Note that each key, $K_{i,j}$, is held by its cluster ($C_{i,j}$) members and its lower level (*children*) subgroups' members.

3.4.1 Initial Hybrid Group Key Management (I-HGKM)

In the following discussion, GDH.2 protocol [67] is used as a building block to generate/regenerate the keying material within each cluster. Note that HGKM protocol can use any key agreement protocol as a building block. In this section, we will explain how HGKM protocol can extend GDH.2 to work efficiently with large size groups.

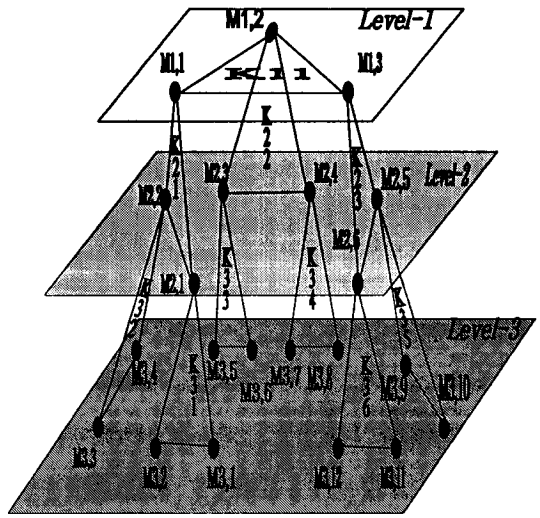

Figure 3.3: HGKM Hierarchical Scheme

The Initial Hybrid Group Key Management protocol, which is depicted in Algorithm 3.1, starts from the lowest level $level_l$. Members of each cluster, $C_{l,x}$ (where $x \in [1, v]$, and $v = d(d-1)^{l-2}$), separately and collaboratively generate their session key $K_{l,x}$ using GDH.2 protocol. This process runs concurrently for all clusters $C_{l,x}$. After agreeing on the cluster session key, $K_{l,x}$, members of each cluster in the upper level, $level_{l-1}$ ($C_{l-1,\acute{x}}$, where $\acute{x} \in [1, \frac{v}{d-1}]$) generate their session keys, $K_{l-1,\acute{x}}$. After agreeing on the cluster session key, each member in the cluster, $C_{l-1,\acute{x}}$, (except the cluster leader $M_{l-2,\acute{x}}$) broadcasts the generated cluster session key $K_{l-1,\acute{x}}$ to its $level_l$ cluster members encrypted by the $level_l$ cluster session key $K_{l,x}$. This process will continue until the cluster leaders in $level_2$, which constitute the root cluster, $C_{1,1}$, generate the $level_1$ session key, $K_{1,1}$, which is the group session key. Then each member in $C_{1,1}$ broadcasts the group session key $K_{1,1}$, encrypted with its $level_2$ session key $K_{2,y}$, ($y = 1, 2, \ldots, d$) to its sibling members.

To illustrate this approach, an explanation of the protocol using a simple numerical example is presented, referring to the hierarchical structure depicted in Figure 3.3. Suppose we choose the number of cluster members to be $d = 3$ and the maximum number of levels to be $l = 3$. After the proper cluster construction is done as explained in section 3.3, and every member has been assigned a certain index according to its level and its number in that level, the key agreement protocol starts.

As mentioned before, I-HGKM protocol starts from level 3. Members of each cluster at $level_3$, $C_{3,x}$ (where $x = 1, \ldots, 6$), use GDH.2 to generate their cluster session key $K_{3,x}$. For example, the cluster $C_{3,4}$, which consists of members $\{M_{2,4}, M_{3,7}, M_{3,8}\}$, generates the key $K_{3,4}$. The cluster leader $M_{2,4}$ with the other members in the cluster $C_{2,2}, \{M_{1,2}, M_{2,3}\}$ generate the cluster session key $K_{2,2}$. Each member of $C_{2,2}$ (except $M_{1,2}$) will broadcast key $K_{2,2}$ to level 3 members encrypted by level 3 cluster session keys. For example:

$$M_{2,4} \Longrightarrow \{M_{3,7}, M_{3,8}\} : \mathbf{E}_{K_{3,4}}(K_{2,2}).$$
$$M_{2,3} \Longrightarrow \{M_{3,5}, M_{3,6}\} : \mathbf{E}_{K_{3,3}}(K_{2,2}).$$

At the same time $M_{1,2}$ with $M_{1,1}, M_{1,3}$, which constitutes the root level cluster $C_{1,1}$, generates the group session key $K_{1,1}$ and broadcasts it to the rest of the group members encrypted with the lower level cluster session key. For example:

$$M_{1,1} \Longrightarrow \{M_{2,1}, M_{2,2}, M_{3,1}, M_{3,2}, M_{3,3}, M_{3,4}\} : \mathbf{E}_{K_{2,1}}(K_{1,1}).$$
$$M_{1,2} \Longrightarrow \{M_{2,3}, M_{2,4}, M_{3,5}, M_{3,6}, M_{3,7}, M_{3,8}\} : \mathbf{E}_{K_{2,2}}(K_{1,1}).$$
$$M_{1,3} \Longrightarrow \{M_{2,5}, M_{2,6}, M_{3,9}, M_{3,10}, M_{3,11}, M_{3,12}\} : \mathbf{E}_{K_{2,3}}(K_{1,1}).$$

Algorithm 3.1 *Initial Hybrid Group Key Management Protocol (I-HGKM)*

INPUT: $d, l, \mathbf{E}_k, f(k), GDH.2$

OUTPUT: $K_{1,1} \in \{KeySet(\text{all group members})\}$

1. For $x = l$ *down to* 2 *do the following:*

1.1 Set $v \longleftarrow d(d-1)^{x-2}$

1.2 Set $\{KeySet(M_{x,z|_{z=1,\ldots,v(d-1)}})\} \longleftarrow NULL$

1.3 For members of $C_{x,y}|_{y=1,2,\ldots,v}$ *do the following:*

1.3.1 Use IKA of GDH.2 (Algorithm 2.4) to generate $K_{x,y}$

1.3.2 $\{KeySet(M_{x,w})\} = \{KeySet(M_{x,w})\} + K_{x,y}|_{w\in[(y-1)(d-1)+1,y(d-1)]}\}$

1.4 If $x < l$ *do the following*

1.4.1 $M_{x,w} \Longrightarrow UserSet(K_{x+1}, w) : \mathbf{E}_{f(K_{x+1,w})}(K_{x,y})$

2. For all $M_{1,j}\ |_{j=1,\ldots,d}$ *do the following*

2.1 Use IKA of GDH.2 (Algorithm 2.4) to generate $K_{1,1}$

2.2 $M_{1,j} \Longrightarrow UserSet(K_{2,j}) : \mathbf{E}_{f(K_{2,j})}(K_{1,1})$

***Remark* 3.1** Note that the length of the generated cluster session key depends on the underlying algebraic group $\mathbb{G}$ used by GDH.2. As the length of the elements of this algebraic group should be very large (see Appendix A) , we need a certain function $f(k)$ to extract the required key length of the symmetric encryption scheme $\mathbf{E}_k$ used. For example, if the protocol uses the DES algorithm for encryption, the used function should extract 56 bits to be used as a symmetric key for DES. For example this function can apply hash function $\mathcal{H}(k_{i,j})$, like SHA-1, to get 160 bits from the generated cluster session key and after that it can pick the first 56 bits to be used as a symmetric key.

We now consider the performance characteristics of the I-HGKM protocol based on the metrics discussed in Section 2.4.5. The proposed hierarchical protocol has

the following characteristics for initial key generation, where the maximum number of cluster members is d, the maximum number of levels is l and the number of clusters is N_c.

Rounds	$R = l(d+1) - 1$
Messages	$M = d(N_C + 1)$
Exponentiations	$Ex_T = Ex_c.N_C$
Encryptions	$E_n = N_C - 1$
Maximum Message size	$M_s = d$

3.4.2 Group Membership Changes

In the previous section, we assumed that the group membership is determined in advance and that all group members have been authorized to share in the group communication prior to execution of the protocol. However, due to the dynamic nature of group communication, the proposed protocol must handle adjustments to group secrets after any membership change operation in the underlying group communication system. The notification of any membership changes is supported by the underlying group communication system. The numbering scheme of the remaining members must be updated accordingly.

The membership changes include (as explained in Section 2.4.4) new member(s) joins and/or old member(s) leaves (or getting evicted from) the group. Moreover, due to environmental factors such as network failure, the communication group may be partitioned. Similarly, when network partitions heal, multiple groups may merge into one. In addition to the aforementioned membership operations, periodic refreshes of group secrets are advisable to limit the amount of ciphertext generated with the same key and to recover from potential compromises of members' contributions of prior session keys.

In the following sections, we will explain how the proposed protocol supports the aforementioned operations (join, leave, merge, partition, key refresh).

The protocols discussed assume that every member can determine (unambiguously) an insertion point in the hierarchical structure. In other words, we assume that after checking the eligibility of the new member to join the group, *NA* provides the joining member with its position in the hierarchical structure accompanied by the public parameters used in the protocol. Also, *NA* is responsible for informing the other group members of the identification of the new member who is joining. Similarly, when any member leaves, *NA* is responsible for updating the membership view for all the remaining members.

Note that the key refresh operation is, in fact, a special case of the leave operation, without any member leaving the group, where the root cluster $C_{1,1}$ regenerates a new group session key $\acute{K}_{1,1}$ and broadcasts it to the other group members encrypted with the suitable lower level keys. Moreover, any cluster can refresh its cluster session key and proceed with the protocol as previously mentioned.

HGKM Join Protocol

The main security requirement of member join is the secrecy of the previous group keys with respect to both outsiders and new group member(s). In the proposed protocol this can be achieved as follows (Algorithm 3.2):

A user who wants to join the group initiates the protocol by sending a join request message to the *NA*, who checks the eligibility of the new member to join this group. If the check passes, *NA* replies with the needed cryptographic public parameters C_P (α and p in case of using GDH.2 as a building block) as well as the specific insertion point (a specific cluster $C_{i,v}$ and its number in the cluster). At the same time *NA* sends to the members of the cluster, where the new member will join,

a message to notify them that there is a new member that will join the group and provides them with its information (the index of the new member which specifies its position in the cluster).

A new member can join at any level which keeps the hierarchical balance i.e, if there is any cluster that contains a number of members $< d$, the new member will join this cluster (to improve protocol performance the preference will be to the upper levels). On the other hand, if the distribution is well balanced then the new member joins at any cluster at any level. Preference will still be given to the upper levels, with the highest preference given to the root cluster $C_{1,1}$ (note that if the members distribution is geographically based, the new member joins the nearest cluster).

After the new member has joined a certain cluster, for example, $C_{i,v}$, the new member index will be $M_{i,v(d-1)+1}$ or $M_{1,d+1}$ in the case of joining $C_{1,1}$. This cluster (including the new member) will regenerate a new cluster session key $\acute{K}_{i,v}$. The old members ($M_{i,x}$) of $C_{i,v}$ (where $x \in [(v-1)(d-1)+1, v(d-1)]$) broadcast the new session key to $level_{i+1}$ clusters $C_{i+1,x}$, encrypted with the keys $K_{i+1,x}$. At the same time, the parent level cluster, $C_{i-1,z}$, ($z = \lceil \frac{v}{d-1} \rceil$) will regenerate a new cluster session key, $\acute{K}_{i-1,z}$, and also broadcast the new session key $\acute{K}_{i-1,z}$ to the lower level clusters. This procedure will continue until the root level cluster key $\acute{K}_{1,1}$ is regenerated and broadcast to the other members. We have to note that all the down broadcasting messages to the $children$ should be encrypted with $Keyset(children)$.

Algorithm 3.2 *HGKM Join Protocol*

1. $M \Longrightarrow NA$: JOINREQUEST
2. $NA \Leftrightarrow M$: AUTHENTICATE
3. $NA \Longrightarrow M : i, v, C_p$
4. *Members of* $C_{i,v}$ *Use AKA of GDH.2 to regenerate* $\acute{K}_{i,v}$
5. *For* $t = (v-1)(d-1)+1$ *up to* $v(d-1)$ *do*
   - 5.1 $M_{i,t} \Longrightarrow Userset(K_{i+1,t}) : \mathbf{E}_{f(K_{i+1,t})}(\acute{K}_{i,v})$
   - 5.2 *For* $x = i-1$ *to* 1 *and* $y_t = \lceil \frac{v}{(d-1)^t} \rceil$ *where* $t \in [1, l-i+1]$
     - 5.2.1 *Members of* C_{x,y_t} *Use AKA of GDH.2 to regenerate* $\acute{K}_{x,y_t}$
     - 5.2.2 *For* $j = (y_t-1)(d-1)+1$ *to* $y_t(d-1)$ *do*
       - 5.2.2.1 $M_{x,j} \Longrightarrow Userset(K_{x,j}) : \mathbf{E}_{f(K_{x,j})}(\acute{K}_{x,y_t})$

From the previous description of the join protocol, it is clear that the overhead cost of the protocol (communications and computations), needed to regenerate the session key, depends on the position where the new member joins. The minimum cost of regeneration will occur if the new member joins at the root level cluster. On the other hand, if the new member joins at level l, it will cost the maximum number of operations.

The following characteristics are the maximum values, i.e if the new member joins any cluster at $level_l$.

Rounds	$R = 2l$
Messages	$M = 2 + d(l-1)$
Exponentiations	$Ex_T = d + 3 + l(2d-1)$
Encryption	$E_n = d + (d-1)(l-2)$
Maximum Message size	$M_s = d$

HGKM Leave Protocol

Leave protocol is initiated in two cases: (1) When a member wishes (voluntarily) to leave the group session, or (2) The *NA* wants to expel a member from the group (maybe due to a certain policy or security compromise). The main security requirement of member leave (especially if it is forced to leave) is the secrecy of the subsequent group keys with respect to both outsiders and former group members. After granting the leave request, *NA* updates the numbering scheme in the cluster $C_{i,v}$ where the member $M_{i,j}$ ($v = \lceil \frac{j}{d-1} \rceil$) leaves by deleting the user $M_{i,j}$ and for all the clusters $C_{x,y}$ (where $x \in [i+1, l]$, and $y \in [j, j(d-1), j(d-1)^{(l-i-1)}]$) who are children of this user; the last member $M_{x,y(d-1)}$ in each cluster replaces its cluster leader $M_{x-1,y}$. Figure 3.4 shows how the hierarchy has been updated after member $M_{1,3}$ leaves the group. It should be clear that there is no need for updating the numbering scheme if the leaving member, $M_{l,j}$ is from any cluster at the $level_l$.

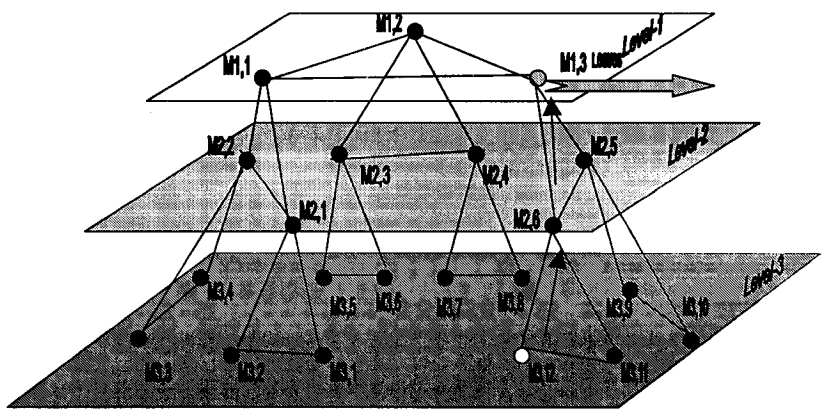

Figure 3.4: The Hierarchy Updating after the Departure of Member $M_{1,3}$

After the hierarchy has been updated, the cluster $C_{i+1,j}$ regenerates a new cluster session key $\acute{K}_{i+1,j}$ and broadcasts it to its lower level cluster members. At the same time the upper clusters $C_{i,v}, C_{(i-1),\lceil \frac{v}{d-1} \rceil} \ldots, C_{1,1}$ regenerate new cluster session keys $\acute{K}_{i,v}, \acute{K}_{i-1,\lceil \frac{v}{d-1} \rceil}, \ldots, \acute{K}_{1,1}$ in sequence and also broadcast these keys (also in sequence) to the child members each encrypted with the associated child's key. The leave protocol is summarized in Algorithm 3.3 and explained using a numerical example in Figure 3.5.

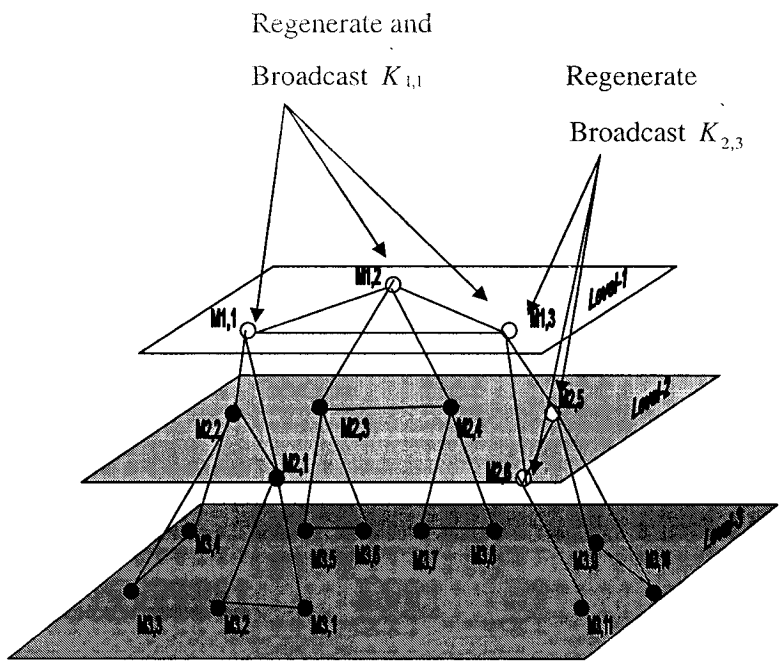

Figure 3.5: Key Regeneration After the Departure of $M_{1,3}$

Algorithm 3.3 *HGKM Leave Protocol*

1. $M_{i,j} \Longrightarrow NA$: LEAVEREQUEST
2. $NA \Longrightarrow M_{i,j}$: LEAVEGRANTED
3. *If* $i < l$ *do the following:*
   - 3.1 *For* $x = i + 1$ *up to* l *do*
     - 3.1.1 $M_{x-1,j} \longleftarrow M_{x,j(d-1)}$
     - 3.1.2 *Set* $j \longleftarrow j(d-1)$
4. *Set* $j \longleftarrow \frac{j}{d-1}$
5. *For* $x = l$ *down to* 2
   - 4.1 *Members of* $C_{x,j}$ *Use AKA of GDH.2 to regenerate* $\acute{K}_{x,j}$
   - 4.2 *If* $x < l$ *do the following:*
     - 4.2.1 *For* $z = (j-1)(d-1)+1$ *up to* $j(d-1)$ *do*
       - 4.2.1.1 $M_{x,z} \Longrightarrow Userset(K_{x+1,z}) : \mathbf{E}_{K_{x+1,z}}(\acute{K}_{x,z})_{K_{x+1,z}}$
       - 4.2.1.2 *Set* $j \leftarrow \frac{j}{d-1}$)
6. *For all* $M_{1,j} \mid j = 1, ..., d$ *do the following*
   - 6.1 *Use AKA of GDH.2 to generate* $K_{1,1}$
   - 6.2 $M_{1,j} \Longrightarrow UserSet(K_{2,j}) : \mathbf{E}_{f(K_{2,j})}(K_{1,1})$

The characteristics of the leave protocol are summarized as following:

Rounds	$R = 2l - 1$
Messages	$M = (l-1)d + 1$
Exponentiations	$Ex_T = l(2d-3)$
Encryptions	$E_n = l(d-1) - d + 1$
Maximum Message Size	$M_s = d$

HGKM Partition Protocol

The partition operation removes m members from the current group of n members. If the partition occurs due to a network failure, there are two cases: 1) The removed members belong to the same cluster, or in other words these members are children of the same parent. In this case, *NA* deals with this situation as if just one member (the parent) leaves. If the parent remains in the group it will regenerate a new key as if this parent has joined the group. 2) These m members do not belong to the same parent. In this case, *NA* can determine whether the remaining $n - m$ members need to construct a new hierarchy or keep the same hierarchy with regeneration of cluster session keys of the remaining clusters with the needed modification to the members' indexing. If the partition is due to multiple leave requests, it will perform the (previously mentioned) leave protocol m-times. As shown in the previous discussion, in the case of partition, the performance of group key agreement protocol using clustering approach is better than that using flat settings. Since there is high probability that the members leaving belong to the same cluster. This makes the cost of the partition protocol for m members the same as a single member leave.

HGKM Merge Protocol

The merge operation is used to add $m > 1$ members to the current group of n members. If $m < d$ it can be treated as m join protocols from the point of view of the cluster where the m members will join. But with respect to the other clusters it will be considered as one member join. Note that using GDH.2 [5], the upflow rounds will be repeated m-times to update the intermediate data and the last round, wherein the new member $d + m$ will be the last member in the cluster and performs just once. If $m \geq d$, the m members can construct a new cluster(s) with their cluster session key(s) and one member can join an old cluster. So with respect to the old n

members, this will be considered as one member request to join the current group. The indexing of the $n + m$ members needs to be updated by NA.

HGKM Key Refresh Protocol

Key refresh protocol is required when group members need to update the group session key, which may be periodically or due to any detected intrusion. In both cases key refresh protocol is initiated by *NA*. Also individual clusters may need to update their session keys. This protocol runs the same procedure as the leave protocol except with $m = 0$. In this case, the cluster will regenerate its new cluster session key and then broadcast it encrypted with its lower level clusters' session keys.

3.4.3 Discussion

In the previous sections, a description of HGKM protocol and its related auxiliary protocols are presented. By adopting clustering techniques, the efficiency of HGKM protocol is improved. A complete complexity analysis of HGKM is given in Section 3.6. HGKM can be considered a hybrid protocol, where both key agreement and key transport are adapted. All the cluster members collaboratively generate the cluster session key, while the cluster leader works as a Key Distribution Center (KDC) and securely distributes the upper level cluster session key to its cluster members.

HGKM has many advantages and disadvantages as well. The first advantage of the protocol is its efficiency and distribution of load between the group members. This feature reduces the time required to generate the group session key. Secondly, robustness against failure since each cluster generates its cluster session key independent of the others; any failure (link or member) within the cluster affects only

the cluster members. The other group members can follow the protocol and generate the group session key. This cluster can join the group later by performing a member HGKM join protocol (Algorithm 3.2). Another advantage of HGKM protocol is its suitability for heterogeneous systems. Some systems have different power capabilities or function as receive only devices. These systems with certain power constraints (like mobile devices) can construct one or more clusters and the cluster leader will be responsible for generating and distributing the group session key. In this case, choosing the cluster leader depends on its power capability. Using this feature, the whole group can accommodate many systems with different characteristics. Another advantage of this hierarchical structure is that it can be used to provide multi-level security. Some military applications classify their documents (during storage or transmission) into different security levels (classified, secret, top secret,...). HGKM can be modified to let each level have a certain security level, where the upper level members can access the lower level documents but not vice versa. Also the direction of communication (up or down) can be controlled by the cluster leader.

Although HGKM is a distributed protocol, it is not a role-symmetric protocol. The cluster leaders take more responsibility than other cluster members. Moreover, the level of dependence on and trust of the cluster leader increases as one goes up the hierarchy towards the root cluster, where one d^{th} of the members should trust one member from the root cluster. From the security point of view, transmission of the upper level cluster keys and the group session key, is not desirable even if they are encrypted. Compromise of one or more keys leads towards a compromise of the group session key. Moreover, the authentication between group members is transitive through the cluster leader. In the following section, we will propose an alternate protocol that can address these concerns.

3.5 Cluster Group Key Agreement

In this section, another scheme that provides a scalable key agreement protocol, Cluster Group Key Agreement protocol (CGKA), will be presented. CGKA is also based on multi-level clustering of the universal group into small, size-bounded clusters. The proposed protocol requires clustering of the group members into a hierarchical structure as shown in Section 3.3. Although CGKA protocol is based on the distribution of group members into (approximately) equal-size clusters, all the members contribute equally in generating the group session keys, which is a major requirement in securing dynamic *peer* groups. The main advantage of this protocol over the HGKM protocol (Section 3.4) is that the group session key and the intermediate clusters' session keys are never transmitted through the network. Furthermore, the members of a cluster can communicate securely within the cluster by either using another cluster session key (wherein the cluster representative adds a different secret value to the contributions of the preceding members and broadcasts the intermediate values to the cluster members only) or by applying a strong hash function to its cluster session key. The construction of the hierarchical structure is the same as in the HGKM protocol but with a different indexing scheme as shown in Figure 3.6.

In the following discussion, $\{M^j_{i,v}\}_s$ represents the descendants of member $M^j_{i,v}$ and can be calculated iteratively as follows:

$$\{M^j_{i,v}\}_s = \{M^x_{i+1,(d-1)(v-1)+j}; x = 1, \ldots, d-1\},$$
$$\{\{M^x_{i+1,(d-1)(v-1)+j}\}_s; x = 1, \ldots, d-1\}$$

3.5.1 Initial Session Key Generation

The initial key generation protocol (Algorithm 3.4) takes place once the hierarchical structure has been created. The main protocol consists of l-steps and each step

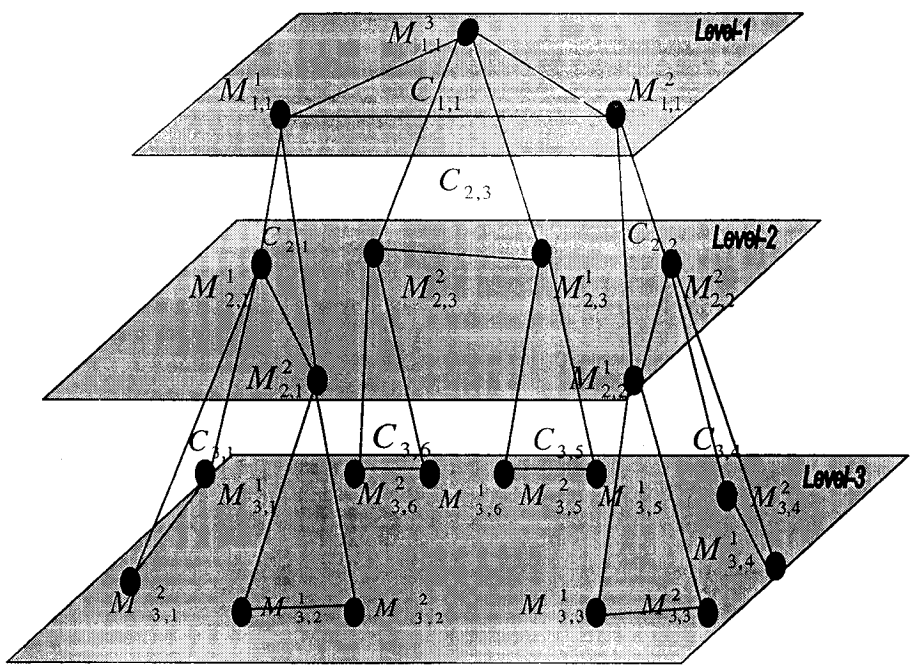

Figure 3.6: Hierarchical Structure of Group Members

comprises of a different number (equal to the number of clusters in this level) of d simultaneous rounds, so that the total number of rounds is $R = ld$. After successfully performing each step, every cluster will be represented by its representative in conducting the next step as shown in Algorithm 3.4, which uses the generated cluster session key as its secret share in conducting the next step.

The protocol starts from the lowest level, $level_l$. For each cluster in $level_l$, $C_{l,v}$ $v = 1, 2, \ldots, d(d-1)^{l-2}$, each member $M^j_{l,v}$, $j = 1, \ldots, d-1$, receives the contributions from $M^{j-1}_{l,v}$ and composes j intermediate values, each with $(j-1)$ exponents and one cardinal value containing j exponents ($M^1_{l,v}$ can be thought of as a cluster initiator, where it uses α as its cardinal value). For example (refer to Figure 3.4) $M^2_{3,2}$ receives $\alpha^{r^1_{3,2}}$ and produces $\{\alpha^{r^1_{3,2}}, \alpha^{r^2_{3,2}}, \alpha^{r^1_{3,2}r^2_{3,2}}\}$. The cardinal value in this example is $\alpha^{r^1_{3,2}r^2_{3,2}}$. By the time the upflow reaches M^y_{l-1,v_1} (the cluster $C_{l,v}$'s representative to the upper level, $level_{l-1}$), the cardinal value becomes $\alpha^{\prod r^j_{l,v}}$. Here, y is equal to the remainder of $(\frac{v}{d-1})$ if $v \nmid (d-1)$, otherwise it is equal to $\frac{v}{d-1}$ if $v \mid (d-1)$ and $v_1 = \lceil \frac{v}{d-1} \rceil$. Upon receiving this message, M^y_{l-1,v_1} adds his secret exponent r^y_{l-1,v_1} to the message. By adding his secret exponent to the cardinal value, M^y_{l-1,v_1} can compute the cluster session key and broadcast the intermediate values, $\{\alpha^{\frac{\prod r^j_{l,v}}{r^j_{l,v}} r^y_{l-1,v_1}}\}$, to its $level_l$ cluster's members $M^j_{l,v}$. All cluster members are now able to generate the cluster session key $\alpha^{\prod\{r^j_{l,v}\} r^y_{l-1,v_1}}$, which will be considered (in the second step) as this cluster's secret share (R^y_{l-1,v_1}). In the second step, which will be conducted by $level_{l-1}$ members, the same procedure within each cluster will be repeated, namely, each member in each cluster sends its public share to the next member in the cluster (which is the upflow rounds) but using the lower level cluster session key (R^j_{l-1,v_1}) as its secret share instead of picking another random value. For example the secret share for $M^1_{2,1} = R^1_{2,1} = \alpha^{r^1_{3,1}r^2_{3,1}r^1_{2,1}}$ and its intermediate (public) value is $\alpha^{R^1_{2,1}} = \alpha^{\alpha^{r^1_{3,1}r^2_{3,1}r^1_{2,1}}}$. In the down flow messages

the cluster's representative (M^y_{l-2,v_2}, where $v_2 = \lceil \frac{v_1}{d-1} \rceil = \lceil \frac{v}{(d-1)^2} \rceil$) broadcasts the intermediate values to all its descendants $\{M^y_{(l-2,v_2}\}_s$. For example (refer to Figure 3.6) $M^1_{1,1}$ broadcasts $\alpha^{r^1_{1,1}\alpha^{(r^1_{3,1}r^2_{3,1}r^1_{2,1})}}$ to $\{M^2_{2,1}, M^1_{3,2}, M^2_{3,2}\}$ and broadcast $\alpha^{r^1_{1,1}\alpha^{(r^1_{3,2}r^2_{3,2}r^2_{2,1})}}$ to $\{M^1_{2,1}, M^1_{3,1}, M^2_{3,1}\}$. This procedure runs from $level_{l-1}$ to $level_2$. At $level_1$ each member uses its $level_2$ cluster session key as its secret share. For instance, in our example, the secret value for $M^1_{1,1}$, $R^1_{1,1} = \alpha^{r^1_{1,1}\alpha^{(r^1_{3,1}r^2_{3,1}r^1_{2,1})+(r^1_{3,2}r^2_{3,2}r^2_{2,1})}}$. Similarly, The secret keys of $M^2_{1,1}$ and $M^3_{1,1}$ are $R^2_{1,1} = \alpha^{r^2_{1,1}\alpha^{(r^1_{3,3}r^2_{3,3}r^1_{2,2})+(r^1_{3,4}r^2_{3,4}r^1_{2,2})}}$ and $R^3_{1,1} = \alpha^{r^3_{1,1}\alpha^{(r^1_{3,5}r^2_{3,5}r^1_{2,3})+(r^1_{3,6}r^2_{3,6}r^2_{2,3})}}$ respectively. As in the previous steps, each member, $M^j_{1,1}$, in cluster $C_{1,1}$ raises the received intermediate values to the power of its private input and forwards the result to the next member in the cluster. The down-flow stage takes place when the last member, $M^d_{1,1}$ receives the last up-flow message and computes the group session key S_n. $M^d_{1,1}$ then raises the intermediate values it has received to the power of its private key $R^d_{1,1}$ ($R^3_{1,1}$ in our example) and broadcasts the intermediate values to the other members, $M^j_{1,1}$; $j \in [1, d-1]$, and their descendants, $\{M^j_{1,1}\}_s$; $j \in [1, d-1]$. At the same time $M^d_{1,1}$ broadcasts its cardinal value (before adding its secret exponent) to its descendants, $\{M^d_{1,1}\}_s$. For example, $M^3_{1,1}$ broadcasts $\alpha^{R^1_{1,1}R^2_{1,1}}$ to its descendants, $\{M^1_{2,3}, M^2_{2,3}, M^1_{3,5}, M^2_{3,5}, M^1_{3,6}, M^2_{3,6}\}$, who have the value $R^3_{1,1}$; broadcasts $\alpha^{R^1_{1,1}R^3_{1,1}}$ to $M^2_{1,1}$ and its descendants, $\{M^1_{2,2}, M^2_{2,2}, M^1_{3,3}, M^2_{3,3}, M^1_{3,4}, M^2_{3,4}\}$, who have the value $R^2_{1,1}$ and broadcasts $\alpha^{R^2_{1,1}R^3_{1,1}}$ to $M^1_{1,1}$ and its descendants, $\{M^1_{2,1}, M^2_{2,1}, M^1_{3,1}, M^2_{3,1}, M^1_{3,2}, M^2_{3,2}\}$, who have the value $R^1_{1,1}$. At this time all the group members are able to generate the group session key, which is: $S_n = \alpha^{R^1_{1,1}R^2_{1,1}R^3_{1,1}}$.

Algorithm 3.4 *CGKA Initial Key Generation Protocol (I-CGKA)*

INPUT: l, d, C_p

OUTPUT: $S_n = \{\alpha^{r^1_{1,1},\cdots,r^d_{1,1}}\}$

1. For $i = l$ *to* 2

1.1 Set $v \leftarrow d(d-1)^{i-2}$

1.2 For $x = 1$ *to* v

1.2.1 For $y = 1$ *to* $d-2$

1.2.1.1 $M^y_{i,x} \Rightarrow M^{y+1}_{i,x} : \{\alpha^{\frac{\Pi\{r^k_{i,x}\}}{r^k_{i,x}}}|_{k\in[1,y]}, \alpha^{r^1_{i,x}\cdots,r^y_{i,x}}\}$

1.2.2 $x_1 \leftarrow \lceil \frac{x}{d-1} \rceil$

1.2.3 If $x \mid d-1$ *, Set* $y_1 \leftarrow \frac{x}{d-1}$

1.2.3.1 Else, $y_1 = Rem(\frac{x}{d-1})$

1.2.4 $M^{y+1}_{i,x} \Rightarrow M^{y_1}_{i-1,x_1} : \{\alpha^{\frac{\Pi\{r^k_{i,x}\}}{r^k_{i,x}}}|_{k\in[1,y+1]}, \alpha^{r^1_{i,x}\cdots,r^{y+1}_{i,x}}\}$

1.2.5 $M^{y_1}_{i-1,x_1} \Rightarrow \{M^{y_1}_{i-1,x_1}\}_s : \{\alpha^{r^{y_1}_{i-1,x_1}\frac{\Pi\{r^k_{i,x}\}}{r^k_{i,x}}}|_{k\in[J,y+1]},\}$

1.2.6 Set $r^{y_1}_{i-1,x_1} \leftarrow \alpha^{r^{y_1}_{i-1,x_1}\Pi\{r^k_{i,x}|k\in[1,y+1]\}}$

2. For $x = 1$ *to* $d-1$

2.1 $M^x_{1,1} \Rightarrow M^{x+1}_{1,1} : \{\alpha^{\frac{\Pi\{r^k_{1,1}\}}{r^k_{1,1}}}|_{k\in[1,x]}, \alpha^{r^1_{1,1},\cdots,r^x_{1,1}}\}$

3. $M^d_{1,1} \Rightarrow \{M^z_{1,1}, \{M^z_{(1,1)}\}_s|_{z\in[1,d-1]}\} : \{\alpha^{\frac{\Pi\{r^k_{1,1}\}}{r^z_{1,1}}|_{k\in[1,d]}}\}$

4. $M^d_{1,1} \Rightarrow \{M^d_{1,1}\}_s : \{\alpha^{r^1_{1,1},\cdots,r^{d-1}_{1,1}}\}$

The previous discussion shows that every group member participates in creating the group session key, which enables each member of the group to assure the freshness of the generated session key. Also, the protocol is very efficient since most of the computation and communication is conducted in parallel over the clusters. Due to this parallelism, we consider only the serial cost of computation.

In summary, CGKA initial key agreement protocol has the following character-

istics:

Rounds	$R = ld$
Messages	$M = dN_C + 1$
Exponentiations	$Ex_S = l[d\frac{d+1}{2} + 1]$
Maximum Message size	$M_s = d$

It should be mentioned that in spite of the number of messages in the CGKA protocol being larger than that of the flat settings (GDH-2, for example), the maximum bandwidth in CGKA protocol is much smaller since the maximum message size is limited to the cluster size, d. Also if any failure happens in the flat setting, the whole protocol halts and must be repeated and all messages should be retransmitted, but in the CGKA protocol a link or member failure can be easily isolated. For example, if a message from any member at any level is belated due to any failure, after a certain time (a specified time out) the whole cluster will be discarded from the group and the protocol completes without this cluster's members, who can join afterwards just as a single member join (the cluster representative). Another advantage is that in a CGKA setting the transmission delay is much smaller than that of the flat setting, since the clustering scheme chooses to distribute the members according to the geographical base. In other words, in the CGKA protocol, clustering is based on a maximum specified distance between all cluster members; the only long distance messages will be between the cluster representatives, therefore, the link failure will affect only inter-representative communications. This is not the case in the flat setting where there may be many transcontinental messages which could take a lot of time and need more reliability, which is not always guaranteed. Also it should be mentioned that in spite the fact that the total exponentiation in CGKA protocol is larger, the average exponentiation (per member) is much smaller than that in the flat setting. Also most of the exponentiations are performed concurrently, which distribute the load between the group members fairly. Lastly, the

special rule of the last member in GDH-2 is distributed in CGKA protocol between the clusters' representatives, which easily distributes the workload and helps in fault isolation. Failure of one representative affects only its cluster (and its descendants), while other group members will not be affected.

As it has been mentioned earlier, the nature of group communications is not static. The membership of the group mutates during the course of the communication. After any membership changes, the session keys should be changed accordingly. The CGKA protocol has efficient auxiliary protocols to support the regeneration of session keys after any membership changes. In the following section, we will introduce the way CGKA supports these changes.

3.5.2 Group Membership Changes

In the following subsections, we will explain how CGKA protocol supports the aforementioned operations, that is join, leave, merge, partition and key refresh protocols. In these subsections, it is assumed that every member can determine (unambiguously) an insertion point in the hierarchical structure. In other words, it is assumed that after checking the eligibility of the new member to join the group, *NA* will provide the joining member with its position in the hierarchical structure accompanied by the public parameters used in the protocol. Also, *NA* is responsible for informing the other group members of the identification of the new joining member. Similarly, when any member leaves, *NA* is responsible for updating the membership view for all the remaining members.

CGKA Join Protocol

In CGKA protocol, the security requirement of a member joining in is the secrecy of the previous group keys with, respect to both outsiders and the new group mem-

ber, which can be achieved as follows (Algorithm 3.5): A user who wants to join the group initiates the protocol by sending a join request message to the network administrator *NA* who checks the eligibility of the new member to join this group. If this check passes, *NA* replies with the needed cryptographic public parameters $C_P(\alpha$ and $p)$ and also the specific insertion point (a specific cluster $C_{l,v}$ and its number in the cluster). Usually, a new member joins the protocol at level l, since the vacant places are always in the lowest level[2]. The insertion point should keep the balance of the hierarchical structure, i.e. the new member joins the cluster with the lowest number of members so as not to increase the levels of the hierarchy. On the other hand, if the group members were distributed in a geographical base, the new member joins the nearest cluster. If the number of members in one cluster becomes too large, it can be split into two or more clusters.

After the join request has been granted, the join protocol begins. Suppose that the new member joins cluster $C_{l,v}$. The cluster's representative, M^y_{l-1,v_1}, who is supposed to keep the upflow message he has received from the previous change, picks another secret exponent, $\hat{r}^y_{l-1,v_1}$, and computes another upflow message and sends it to the new member who is slated to take the role of the cluster representative. Upon receiving the intermediate values, the new member adds its secret share $r^n_{l,v}$ and broadcasts the intermediate values to the other clusters' members. All the cluster members and their descendants are now able to generate their new cluster session key. After regenerating the cluster session key, the (old) cluster representative, M^y_{l-1,v_1}, requests that its ancestor cluster, C_{l-1,v_1}, refresh its session key. M^y_{l-1,v_1} uses the new lower cluster's session key as a new secret share. According to its position in the cluster ($y = 1, 2, \ldots, d-1$), M^y_{l-1,v_1} adds its public share to the message it has received from its preceding member in the last conduction of the

[2]The reason for this choice will be explained in the following subsection, which explains the Leave protocol.

key generation/regenration protocol and sends this message to its next member. By the time the new intermediate values has been received by M^{y}_{l-2,v_2} (C_{l-1,v_1} cluster's representative), he puts its share (which may have changed) and broadcasts these intermediate values to its descendants. This process continues until the root cluster regenerates a new group session key.

Algorithm 3.5 *CGKA Join Protocol*

Consider the new member $M^{n}_{l,v}$ joins at the cluster $C_{l,v}$

1. If $v|d-1$, Set $y \leftarrow \frac{v}{d-1}$

1.1 Else, $y = Rem(\frac{v}{d-1})$

2. Set $v_1 \leftarrow \lceil \frac{v}{d-1} \rceil$

3. $M^{y}_{l-1,v_1} \Rightarrow M^{n}_{l,v} : \{\alpha^{\hat{r}^{y}_{l-1,v}\frac{\Pi\{r^{k}_{l,v}\}}{R^{k}_{l,v}}|_{k\in[1,d-1]}}\}, \alpha^{r^{1}_{l,v},\ldots r^{d-1}_{l,v}}$

4. $M^{n}_{l,v} \Rightarrow M^{y}_{i-1,v_1}, \{M^{y}_{(i-1,v_1)}\}_s : \{\alpha^{\hat{r}^{y}_{i-1,v}r^{n}_{l,v}\frac{\Pi\{r^{k}_{l,v}\}}{r^{k}_{l,v}}|_{k\in[1,d-1]}}\}, \alpha^{r^{n}_{l,v},r^{1}_{l,v},\ldots r^{d-1}_{l,v}}$

5. Set $\hat{r}^{y}_{l-1,v} \leftarrow \alpha^{\hat{r}^{y}_{l-1,v}r^{n}_{l,v}\Pi\{r^{k}_{l,v}|_{k\in[1,d-1]}\}}$

6. For $i = l-1$ to 2

6.1 Set $x_i \leftarrow \lceil \frac{v}{(d-1)^{l-i}} \rceil$

6.2 Each Cluster C_{i,x_i} does the same role as in I-CGKA Prototocol

7. For $x = 1$ to $d-1$

7.1 $M^{x}_{1,1} \Rightarrow M^{x+1}_{1,1} : \{\alpha^{\frac{\Pi\{r^{k}_{1,1}\}}{r^{k}_{1,1}}}|_{k\in[1,x]}, \alpha^{r^{1}_{1,1},\ldots,r^{x}_{1,1}}\}$

8. $M^{d}_{1,1} \Rightarrow \{M^{z}_{1,1}, \{M^{z}_{(1,1)}\}_s|_{z\in[1,d-1]}\} : \{\alpha^{\frac{\Pi\{r^{k}_{1,1}\}}{r^{z}_{1,1}}}|_{k\in[1,d]}\}$

9. $M^{d}_{1,1} \Rightarrow \{M^{d}_{1,1}\}_s : \{\alpha^{r^{1}_{1,1},\ldots,r^{d-1}_{1,1}}\}$

The cost of the join protocol depends on the position of the cluster in the overall hierarchical structure where the new member joins (i.e. the level where this cluster is located and also the positions of the representative of each ancestor cluster up to the root cluster). The following table shows the cost of the join protocol where we

consider that the new member joins at the lowest level l and all the representatives are located in the first position in the cluster numbering, which is the worst case in CGKA setting.

Rounds	$R = 2 + (l-1)d$
Messages	$M = 2 + (l-1)d$
Exponentiations	$Ex_S = (l-1)(d\frac{d+1}{2}) + l + 1$
Maximum Message Size	$M_s = d$

CGKA Leave Protocol

After granting the leave request to a member, *NA* sends a notification message to the other members of her/his cluster to update the numbering scheme of their cluster. This change in indexing scheme will result in replacing the removed member by the last member of its lower level cluster[3]. For example, if member $M^2_{2,2}$ from cluster $C_{2,2}$ leaves the group (Figure 3.6) the last member of the cluster $C_{3,4}$, $M^2_{3,4}$, will take its place. This updating process of member distribution continues until $level_l$, leaving the vacant places always in $level_l$. After the member numbering has been updated, the key regeneration protocol begins. The protocol starts from the lowest level cluster (cluster $C_{3,4}$ in our example). The new cluster representative of this cluster picks another secret share and adds it to the last message it has received from the preceding member in the cluster during the last rekeying protocol (or from the initial key generation, if it is the first membership change in the session course), and broadcasts these new intermediate values to the remaining cluster members. In the next stage, the protocol runs as the initial key generation protocol. The only difference is that the protocol starts from this new representative, who adds the new share to the contributions from its preceding member and forwards the results to the

[3]The updating of the indexing scheme is the same as the one mentioned in HGKM protocol.

next member. This procedure continues until the root layer as is described in the initial key generation Protocol (Section 3.5.1). The main difference between this protocol and initial key generation is that only a subset of the group members will be involved in the key regeneration protocol (namely, in our example the members of $C_{3,4}, C_{2,2}, C_{1,1}$ will be involved), while the other members will not participate until the last round (broadcasting round).

Since only a subset of group members participate in regenerating the new group session key, the cost of performing this protocol is based on the position of the departed member. The worst case happens if the departed member was in the lowest level and the position of the representatives up to the root level are the first members in their clusters. The best case occurs when the member who left was in the root level. The following characteristics show the cost of the leave protocol, when the worst case is considered.

Rounds	$R = 1 + (l-1)d$
Messages	$M = 1 + (l-1)d$
Exponentiations	$Ex_S = 2 + (l-1)(d\frac{d+1}{2})$
Maximum Message Size	$M_s = d$

CGKA Merge Protocol

The merge operation is used to add $m > 1$ members to the current group of n members. As mentioned earlier (Section 3.5.2), new members always join the group at the lowest level. If $m < d$ and if there are empty spaces to accommodate these members then it can be treated as the join operation used m-times from the point of view of the cluster where m members intended to join. But with respect to the other clusters it will be considered as a single member join. If there are no empty spaces to accommodate these new members, they can construct a new cluster and their representative joins the group as a single member join. If the new members join

different clusters at the same time, it will also be considered as the join procedure being conducted m-times. The only difference in this case is that the number of members involved in the protocol will be increased according to the number of clusters where the new members will join. We should note that the cost of the merge protocol is approximately the same as the join protocol, since the joined clusters at the same level perform the required protocol steps concurrently. If $m \geq d$, the m members can construct a new cluster(s) with their cluster session key(s) and one (or more) member (which will be considered as the cluster representative(s)) can join an old cluster as one (or more) member(s). From the point of view of the old group members, the number of joining members in this case is equal to $\lceil \frac{m}{d} \rceil$. So with respect to the old n members, it will be considered as one (or more) member(s) request to join the current group. The indexing of the $n + m$ members needs to be updated by *NA*.

CGKA Partition Protocol

The partition operation removes m members from the current group of n members. If the partition occurs due to a network failure, there are two cases: 1) The members to be removed belong to the same cluster, i.e. these members belong to the same representative. In this case, *NA* will deal with this situation as a single member (the representative) leaving. If the representative remains in the group it will regenerate a new key as if this representative had joined the group or had initiated a key refresh protocol by changing its secret share. 2) These m members do not belong to the same representative, i.e. they belong to different clusters, then *NA* can determine whether the remaining $n - m$ members need to construct a new hierarchy or keep the same hierarchy. In both cases, the remaining members need to regenerate a new cluster session key taking into account the needed modification to the members'

indexing. If the partition is due to multiple leave requests, it will perform the leave protocol m-times.

CGKA Key Refresh Protocol

Key refresh protocol updates the group session key. This protocol is initiated by *NA* and where the key refresh period depends on the application policy. Generally, if the group membership is static for a certain period, the key refresh protocol should be initiated. The key refresh protocol can be considered a special case of the leave protocol where there are no departed members. It starts, like the leave protocol, from the lowest level. The cluster representative of a certain cluster, $C_{l,v}$ (chosen by *NA*) picks a new secret share and reruns the broadcasting round to its cluster members. After generating a new cluster session key, the protocol runs exactly as the leave protocol.

If a certain cluster needs to refresh its cluster session key, the protocol can be runs for this cluster independent of the rest of the group members. In this case, the cluster will regenerate its new cluster session key without changing the group session key. The cost of refreshing the group session key is similar to that of the leave protocol.

3.6 Complexity Analysis

This section analyzes the communication and computation costs for the aforementioned protocols, namely, initial key generation, join, leave, merge, and partition protocols of HGKM and CGKA. A comparison between the proposed protocols and the authenticated contributory group key agreement scheme GDH.2 described in[5, 66] and with TGDH [43] will be presented. The computational complexity

of TGDH is reduced from $O(n)$ to $O(\log n)$, but the communication cost is slightly larger than that of GDH.2.

The overhead of HGKM and CGKA protocols depend on the structure of the hierarchy, i.e. the maximum number of members in any cluster d and the maximum number of levels l in the hierarchy. As we saw from the characteristics of each protocol, there is a tradeoff between d and l. The number of rounds in the initial protocol requires $O(ld)$ for generation of the session key. In the case of membership change protocols, on the other hand, it requires a lower number of rounds compared to the initial key generation protocol (as expected) to regenerate the session key. The number of messages and the computation overhead of the initial protocols depend on the maximum number of clusters, N_C (which is approximately $O(d^{l-1})$), so it will be affected more by increasing the number of levels. In the case of membership change protocols, the number of messages and the computation overhead requires $O(ld)$ to regenerate the session key. To conclude, the number of levels has to be minimized to achieve the best efficiency, but at the same time the maximum number of members in any cluster has to be bounded above so as not to lose the benefits of HGKM and CGKA protocols.

It is clear that the computation overhead of both HGKM and CGKA protocol is superior to GDH.2 and TGDH, as on average it requires $O(ld)$ to generate/regenerate the session key compared to $O(n)$, which is approximately $O(d^l)$ required by GDH.2 (recall that by using a hierarchical structure in both protocols, the group size n can be represented as follows $n = d\frac{(d-1)^l-1}{d-2} \simeq d^l$). With respect to the communication overhead, the number of rounds in both HGKM and CGKA initial key agreement protocol is $O(ld)$, which is lower than that of GDH.2, which requires $O(n)$. The number of messages, on the other hand, is greater than that of GDH.2 but the maximum bandwidth in both protocols is d compared with n in GDH.2. HGKM and CGKA protocol allow parallel handling of the messages through the network.

Operation	Initial Key Generation			
Protocol	GDH.2	TGDH	HGKM	CGKA
R	n	$2h$	$l(d+1)-1$	ld
M	n	$2nh$	$d(N_C+1)$	$d(N_C)+1$
Ex_S	$\frac{n(n+3)}{2}-1$	$2nh$	$E_{xc}(N_C)$	$l[d\frac{d+1}{2}+1]$
M_s	$n(n-1)$	$2n$	d	d

Table 3.1: Initial Key Generation Comparison

As a result, the total load on the network from HGKM and CGKA protocol is less than that of GDH.2.

The communication overhead of TGDH is $2nh$, where h is the height of the tree, which is slightly larger than that n of GDH.2. It is clear that HGKM and CGKA protocols are superior with respect to communication overhead. The computational overhead is approximately the same in both protocols, especially in the case of membership changes, since both protocols are multi-round protocols. But the HGKM and CGKA approaches are more efficient for large size groups. Since the tree in TGDH is binary, the depth of the tree h ($h = \log_2 n$ if the tree is fully balanced) is greater than the number of levels of the HGKM and CGKA protocols which allows both protocols to have a lower number of rounds and exponentiations. Also the number of users involved in the key regeneration is lower in HGKM and CGKA protocols. A summary of the comparisons are listed in Tables 3.1, 3.2, 3.3.

3.7 Similarities With Other Approaches

Scalability of key distribution protocols have been considered using different approaches (see for example [6, 43, 53, 74]). The proposed framework has similari-

Operation	Join Protocol			
Protocol	GDH.2	TGDH	HGKM	CGKA
R	2	6	$2l$	$2 + d(l-1)$
M	2	2	$2 + d(l-1)$	$2 + d(l-1)$
Ex_S	$3n+2$	$\frac{3h}{2}$	$d + 3 + l(2d-1)$	$(l-1)(d\frac{d+1}{2}) + l + 1$
M_s	n	$2nd$	d	d

Table 3.2: Join Protocol Comparison

Operation	Leave Protocol			
Protocol	GDH.2	TGDH	HGKM	CGKA
R	1	1	$2l-1$	$1 + d(l-1)$
M	2	2	$2 + d(l-1)$	$1 + d(l-1)$
Ex_S	$3n+2$	$\frac{3h}{2}$	$l(2d-1)$	$2 + (l-1)(d\frac{d+1}{2})$
M_s	n	$2n$	d	d

Table 3.3: Leave Protocol Comparison

ties with some of these protocols. For example Iolus protocol [53] uses clustering to distribute the group members into small clusters. Iolus protocol neither discusses suitable clustering techniques nor which criteria it can use to apply the clustering scheme. Moreover, the protocol does not explain a way to generate a global group session key and requires intermediary cluster controllers to transfer intra-cluster communications and to perform key translations. Another example is TGDH protocol [43] where the members are distributed in a binary tree. As shown in Section3.6, the performance of HGKM and CGKA are better than that of TGDH. Furthermore, each member should store a number of intermediary keys equal to the tree height h. STR [64] can be considered a special case of TGDH with a special tree structure, but the generated key has the same format as that of TGDH.

3.8 Summary

In this chapter, a novel framework for a scalable and fault-tolerant key agreement protocol has been presented. The most important feature of HGKM and CGKA protocols is the adoption of a clustering structure in key agreement protocols. Using clustering, we were able to reduce the cost of session key generation/regeneration and easily isolate any member or link failure away from the other group members. Also, we could support group communication with heterogenous systems. This allowed us to develop a solution to the scalability problem in key agreement protocols. The used approach is an efficient and scalable modified extension of the well known DH key exchange protocol to a multiparty key agreement protocol. Although, a modified GDH.2 is used as a building block to generate a cluster session key in the HGKM protocol and the CGKA protocol follows the same line in collecting the members contributions, the generated group session key is different than that of GDH.2. The efficiency of both protocols is also independent of the building

block protocol used to generate the cluster session key.

Although HGKM and CGKA were introduced as a solution to the key agreement problem, both protocols offers comparable advantages to the best known key distribution protocol. Moreover, the storage requirement for each of the members, using both the HGKM and CGKA protocols are superior to those using key distribution. In fact, the group members in CGKA protocol do not have to store intermediate keys to generate/regenerate the session key.

Chapter 4

Group Key Agreement for Ad Hoc Networks

In this chapter, a modification to the framework presented in the previous chapter is proposed. The modified protocol is suitable for different applications that use ad hoc wireless networks as a means of communication. The proposed protocol allows authorized group members to generate their session key without the need of a trusted third party. Also, it does not require a fixed infrastructure, that needs either centralized services or a fixed topology.

4.1 Introduction

Ad hoc wireless networks have become an integral part of all kinds of network infrastructures. They typically are communication networks that usually do not have a pre-existing infrastructure and consist of mobile terminals that connect by relaying messages from one point to another via peer devices. This causes links to be unreliable and the network topology to be dynamic. Additionally, the involved par-

ties may not have a common history. The lack of infrastructure implies an absence of central network management functionality, fixed routers, name servers and much more. This lack of structure in ad hoc networks makes them more vulnerable to attacks than structured networks. This is in addition to the already existing weakness in wireless networks due to the use of radio waves as the common communication medium which make them easily accessible through the use of the right kind of radio.

Ad hoc wireless networks find applications in military operations in which aircraft, tanks, and moving personnel can communicate through mobile devices. Rescue missions and emergency circumstances are also situations where applications for such networks are beneficial. Other examples include virtual classrooms and conferences wherein people can set up a network on the spot through their laptops, PDAs, or other mobile devices assuming that they are sharing the same physical medium.

In such spineless networks, the exchange of cryptographic keys may have to be addressed on demand and without assumptions about a priori negotiated secrets. There are many proposed key agreement protocols designed for wireless networks based on private-key cryptography. The reason for preferring private-key cryptosystems is that the wireless devices are not fully qualified to perform the heavy computations required in public-key cryptosystems. However, private-key based solutions require an on-line trusted third party (TTP) to distribute the session keys. On the other hand, there are many public-key based protocols in the literature for group key establishment (refer to Section 2.5) including the protocols presented in the previous chapter, namely HGKM and CGKA. These protocols have not been designed with the special nature of ad hoc networks in mind. Moreover, these protocols require many modular exponentiations, which is computationally costly and may not be suitable with current technology.

In this chapter, we first present some of the attributes of ad hoc wireless networks as well as a brief survey of the related ad hoc network security mechanisms. A modified key agreement protocol, for a group of communicating nodes in an ad hoc wireless network, is presented. We also give an authenticated version of our protocol based on using identity based pairwise key for entity authentication. This approach has an advantages of not requiring the use of CA. Furthermore, by using a hierarchical structure, the number of required computations have been significantly reduced, making it suitable for resource-constraint devices in ad hoc networks.

4.2 Attributes of Wireless Communication Systems

One of the major challenges in ad hoc network security is that ad hoc networks typically lack a fixed infrastructure both in the form of physical infrastructure such as routers, servers and stable communication links and in the form of an organizational or administrative infrastructure. Another difficulty lies in the highly dynamic nature of ad hoc networks since new nodes can join and leave the network at any time. The major problem in providing security services in such infrastructure-less networks lies on how to manage the cryptographic keys that are needed.

When designing protocols for ad hoc networks, whether routing protocols or security protocols, it is important to consider the characteristics of the network and realize that there are many "flavours" of ad hoc networks. Ad hoc wireless networks generally have the following characteristics [36]:

- Dynamic network topology: The network nodes are mobile and thus the topology of the network may change frequently. Nodes may move around within the network but the network can also be partitioned into multiple smaller networks or be merged with other networks.

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

- Energy constrained nodes: Nodes in ad hoc networks most often rely on batteries as their power source. The use of computationally complex algorithms may not be possible. This also exposes the nodes to a new type of denial of service attack and sleep deprivation torture attack [36] that aims at depleting the nodes energy source.

- Limited physical security: The use of wireless communication and the exposure of the network nodes increase the possibility of attacks against the network. Due to the mobility of the nodes the risk of them being physically compromised by theft, loss or other means will probably be greater than that for traditional network nodes.

In many cases the nodes of ad hoc networks may also have limited CPU performance and memory, e.g. low-end devices such as PDAs, cellular phones and embedded devices. As a result, certain algorithms that are computationally or memory expensive might not be applicable.

4.3 Threats and Vulnerabilities in Wireless Networks

In this section, the concentration will be on wireless networks using the radio path as the transmission medium. The basic security needs are similar to those for wired networks. However, problems related to these needs are stressed in wireless networks due to the use of the radio path. Some of the security threats related to wireless networks can be listed as follows:

- Passive eavesdropping is very easy in radio environment. When one sends a message over the radio path everyone equipped with a suitable transceiver in the range of the transmission can eavesdrop on the manipulated messages. The sender or intended receiver has no means of knowing if the transmission has been overheard or not, so this kind of eavesdropping is absolutely undetectable. In wireless environments the ease of eavesdropping justifies quite costly procedures to guarantee the confidentiality of the network traffic.

- When a wireless network becomes a part of a wired large network it offers one interface to the attacker, requiring no physical arrangements, to intrude on the whole network. This attack is known in the literature as a *transitive trust attack* since if the attacker can fool the wireless part of the network to trust the mobile he controls, in which case it is very difficult to prevent any hostile actions taken by the adversary. This makes the need for efficient authentication mechanisms crucial for the security of wireless networks. In all cases all parties of the transmission should be able to authenticate each other.

- Due to the nature of radio transmissions, wireless networks are vulnerable to denial of service attacks. If the attacker has a powerful enough transceiver, he can easily generate radio interference so that the wireless network is unable to communicate using radio paths. Protection against this kind of attack is very difficult and expensive. The only complete solution is to have the wireless network physically secure, but this is applicable only in very rare cases. It is easy however, for authorities to locate the transceiver used to generate interference, so the attacker has limited time before the transceiver is found.

4.4 Ad Hoc Networks Security

Ad hoc wireless network security research often focuses on secure routing protocols, which form an essential component of ad hoc wireless network security. However, all such protocols assume that key distribution has taken place or in the best cases they are partially described (see for example, SEAD [35], Ariadne [34], ARAN [26], SPINS [61]). Only recently there have been some attempts to define the key distribution problem in ad hoc networks as discussed below.

In [41], Khalili *et al.* have suggested a key distribution protocol that combines Identity-Based Encryption (IBE) and threshold cryptography to be used in ad hoc networks.. The proposed protocol does not address clearly how the nodes involved would authenticate each other in the initialization stage of their proposal, which can create an opportunity for impersonation attacks right from the beginning. Moreover, there is no concrete method given for generating the master key (for both public and private master keys). Zhou and Hass [76] introduce the idea of distributing a certificate authority (CA) through the network in a threshold fashion, at the time of configuring the network. However, the authors do not address the resource limitations of devices in ad hoc networks as public-key and threshold cryptography are (in general) computationally expensive and need to be tailored to the resources and constraints of low-power devices. In [46], the authors propose acceleration techniques for the key establishment protocols using techniques that involve the assistance of a base station called Server-Aided Secret Computation (SASC). In their scheme, SASC is responsible for exchanging information with the base station to ensure that all expensive computations are carried out by the server. In such protocols, a prior arrangement of the base station is required and restricts the independence of nodes in return for assistance from the base station. A password-based authenticated key exchange protocol has also been proposed by

Askon and Ginzboorg in [4].

4.5 Dynamic Topology GKA Protocol (DTGKA)

In this section, a framework for an efficient and scalable key agreement protocol, Dynamic Topology Group Key Agreement protocol (DTGKA) is presented. The proposed protocol is a modified version of the CGKA protocol [3] presented in Section 3.5, which is based on multi-level clustering of the universal group into small, size-bounded clusters.

CGKA was primarily proposed for a group of members with a static topology. Since the nature of ad hoc wireless networks is dynamic and there is no static structure or topology, CGKA is not suitable for ad hoc networks. In the following subsections, we explain the way CGKA can be modified to be suitable for ad hoc networks.

The proposed DTGKA protocol comprises of two phases:

1. Organization of the network nodes into clusters in a multi-level structure.
2. Generation of the group session key.

4.5.1 Initial Setting

DTGKA protocol considers that each node is equipped with a secret *group identity key* K_{IG}, a one-way hash function $\mathcal{H}$, a strong private-key encryption algorithm $\mathbf{E}$ and certain public parameters for the key generation protocol. The group identity key guarantees the authenticity of the node as a member of a group, not its individual identity. Public parameters used are p, q, $\mathbb{G}$ and α, where p is a prime integer and q a prime divisor of $p - 1$. Let $\mathbb{G}$ be the unique cyclic subgroup of $\mathbb{Z}_p^*$ of order q, and α be a generator of $\mathbb{G}$.

The protocol also considers that each node has its local identifier (ID) and has the ability to compute its *weight*. The node weight is a numerical quantity which expresses the current status of the node. There are many factors which affect calculation of the node weight: mobility, battery power level, distance from the other nodes, values related to the surrounding environment (terrain, temperature, battery power, etc.) [8]. Since the main concern of the thesis is the protocol efficiency and since some nodes, in DTGKA protocol, will be required to perform more computation than others, the factors which affect the computational capability of the node will only be considered.

4.5.2 Clusters Construction Phase

There are some constraints that should be taken into account while designing a clustering scheme. First, we have to make sure that all clusters have minimum and maximum size constraints. A maximum size constraint limits the cluster size to which effectively improves the protocol performance. Ideally, we want all clusters to be of the same size so that no clusters are overburdened, or under-burdened, by the processing and storage requirements of the proposed key agreement protocol, although this may not be a realistic assumption for ad hoc wireless networks. Our protocol is flexible enough that it only requires the cluster size to be limited to a given but arbitrary size and allows for the height of each branch of the structure to be variable.

In the first step, each node makes its active neighbours aware of its presence by broadcasting an initial IAMALIVE message, containing its ID and weight encrypted with $K^i_{\mathcal{H}} = \mathcal{H}(S^{i-1}_K)$, i.e. the key obtained from applying the one-way function $\mathcal{H}$ to the key generated from the previous key regeneration. Initially, $S^0_K = K_{IG}$.

Once the nodes have gathered information about their neighbours, the second

step begins (cluster construction): The heaviest node within a certain region (one-hop away neighbours) broadcasts another message which will be treated only by the heaviest node in each region. By receiving this message, the most heaviest d nodes declare themselves as the whole group leaders and assigns a certain number to each one based on its weight within the leaders. For example (see Figure 4.1), the lightest weight node will be indexed as M_1 and the heaviest weight node will be indexed as M_d. Those nodes ($M_i; i = 1, \ldots, d$) constitute the root cluster, C_0. Each member, M_i of the root cluster will declare itself as a leader. Then, it broadcasts to its 1-hop neighbours, except C_0 members, a message, IAMLEADER. A node receiving IAMLEADER message replies with IAMCHILD message to its leader. The leader confirms its leadership to the first $d-1$ children accompanied with an indexing to each node related to its weight. For example the lightest weight node will be the first node in the indexing scheme, $M_{i.1}$ and the second lightest one will be assigned as $M_{i.2}$ and so on. These children mark themselves as a child to this leader and constitute cluster C_i. Note that the cluster size may vary with the upper limit of $d-1$. The process continues for the members of the second level clusters, namely each member, $M_{i.j}$, where $j = 1, \ldots, d-1$, broadcasts to its 1-hop neighbours, except its cluster members and its ancestor, IAMLEADER message. A node receiving IAMLEADER message replies with IAMCHILD message to its leader. The leader confirms its leadership to the first $d-1$ children accompanied with an indexing to each node related to its weight as previously mentioned. Note that if this member does not receive any reply within a certain period, its status will not change to a leader but continue as a child; this condition applies to all members including C_0 members. This process continues until children do not get any replies which means that all members have been assigned a place in a certain cluster. The previous construction can be seen as considering each member of the root cluster as a root member of a different tree. As a result, we have d subtrees for a multi-rooted tree.

The height of the d subtrees can vary according to the distribution of the members in the network, even within the same subtree. If the distribution of the members is uniformly distributed within a certain cluster size, we can say that our hierarchical structure is a well-balanced structure with d-root tree with $(d-1)$-ary subtrees each with maximum height h_i.

Calculations in a well-balanced multi-root $(d-1)$-ary tree: The total number of group members (n) can be represented as a function of the maximum number of members per cluster (d) and the height of each subtree (h) as follows:

First we need to calculate the number of members in each subtree as a full balanced $(d-1)$-ary tree as follows:

$$n_j = \sum_{i=1}^{h}(d-1)^{i-1} = \frac{(d-1)^h - 1}{d-2}. \tag{4.1}$$

By summing the total number of members in all subtrees (n_j), where $j = 1, \ldots, d$ we can calculate the maximum number of members in the group as follows:

$$n = n_1 + n_2 + \ldots + n_d. \tag{4.2}$$

If h_j are equal in all subtrees, which will be considered as a perfectly balanced structure, the maximum number of members can be expressed as:

$$n = dn_j = d\frac{(d-1)^h - 1}{d-2}. \tag{4.3}$$

Note that, in the following discussion, we will use the universal addressing as a numbering scheme for the group members. The address of a member in a rooted subtree is a sequence, for example if a member x in level $h \geq 1$ has children, their addresses are $x.1, x.2, \ldots$. We define $ch(x)$ as the number of x's children. If $ch(x) = 0$, then x is a member with no children, i.e., it confirms itself as a child. If all the cluster members (except the cluster leader) are children, this cluster will be refereed as a leaf cluster. Also, we define $L(x)$ as a leader of the member x. $L(x) = 0$ indicates that x is a member of the root cluster.

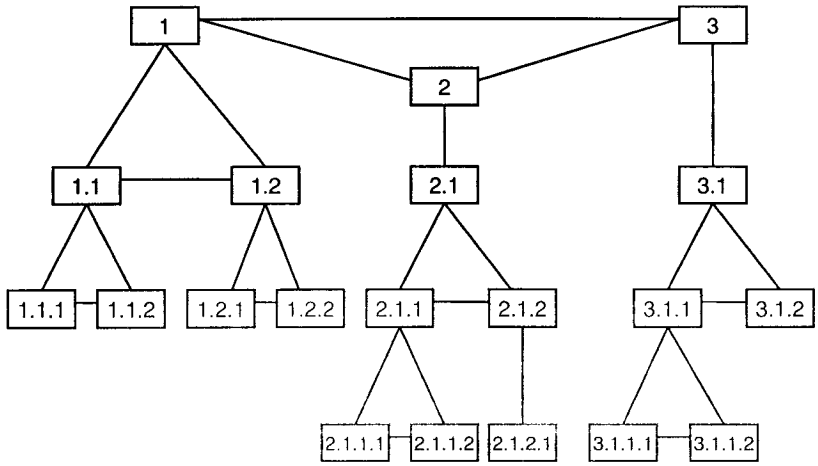

Figure 4.1: Distribution of Group Members

4.5.3 Key Agreement Phase

The protocol starts after distribution of all the authorized members into clusters has been taken place. It should be pointed out that the result of the protocol is a group session key which is accessible by all group members. Furthermore, the members of a cluster can communicate securely within the cluster by either using another cluster session key (wherein the cluster leader adds a different secret value to the contributions of its children and broadcasts the intermediate values only to its children) or by applying a strong hash function to its cluster session key.

Session Key Generation

The key generation protocol (Algorithm 4.1) takes place once the hierarchical structure has been created. In other words, when the members do not receive any replies from their neighbours and confirm themselves as children. This case can be reached in each subtree independently, which means that subtrees can start the protocol from different points. The first phase of the protocol is the same for all subtrees, so the

steps of the protocol through one subtree will be explained and the same can be applied to others. The second phase of the protocol starts once each subtree ends up with a certain secret value known by all the members of the subtree.

For each subtree: the lightest weight child in each leaf cluster, $M_{i.j.k.....1}$ initiates the protocol by picking a random value $r_{i.j.k.....1} \in \mathbb{Z}_P^*$, calculates $\alpha^{r_{i.j.k.....1}}$ (note that all the calculations in the protocol are modulo p, as explained in Section 4.5.1) and sends the result to its neighbour in the cluster, $M_{i.j.k.....2}$, here i is the index of its root member, $i = 1.....d$ and $j, k, l, ...$ are indices of its leaders in the path to the root member, $j, k = 1.....d-1$. Each member $M_{i.j.k.....l}$ receives the contributions from $M_{i.j.k.....l-1}$ and composes j intermediate values, each with $(j-1)$ exponents and one cardinal value containing j exponents ($M_{i.j.k.....1}$ can be thought of as a cluster initiator, which uses α as its cardinal value). For example, as shown in Figure 4.1, $M_{1.2.2}$ receives $\alpha^{r_{1.2.1}}$ from $M_{1.2.1}$ and produces $\{\alpha^{r_{1.2.1}}, \alpha^{r_{1.2.2}}, \alpha^{r_{1.2.1}r_{1.2.2}}\}$. The cardinal value in this example is $\alpha^{r_{1.2.1}r_{1.2.2}}$ by the time the upflow reaches $M_{1.2}$ (the cluster $C_{1.2}$'s leader). Upon receiving this message, $M_{1.2}$ adds its secret exponent $r_{1.2}$ to the message. By adding its secret exponent to the cardinal value, $M_{1.2}$ can compute the cluster session key and broadcast the intermediate values $(\alpha^{r_{1.2.1}r_{1.2}}, \alpha^{r_{1.2.2}r_{1.2}})$ to its children, $M_{1.2.1}$ and $M_{1.2.2}$. All cluster members are then able to generate the cluster session key $\hat{r}_{1.2} = \alpha^{r_{1.2}r_{1.2.1}r_{1.2.2}}$.

In the second step, which will be conducted by $level_{h-1}$ members, the same procedure within each cluster will be repeated, namely, each member in each cluster sends its public share to the next member in the cluster but using the lower level cluster session key as its secret share instead of picking another random value. For example the secret share for $M_{1.2} = \hat{r}_{1.2}$ and its intermediate (public) value is $\alpha^{\hat{r}_{1.2}} = \alpha^{\alpha^{r_{1.2}r_{1.2.1}r_{1.2.2}}}$. In the down flow messages, the cluster's leader broadcasts the intermediate values not only to its children but to all its descendants. This procedure runs from $level_{h-1}$ to $level_2$. The second phase starts when all the members in

the root cluster are able to share a secret value with all its descendants. Each member in the root cluster uses this secret value as its secret share in the second phase when performing the last part of the key agreement protocol. For instance, in our example the secret value for M_1, $\hat{r}_1 = \alpha^{r_1\alpha^{\hat{r}_{1.1}+\hat{r}_{1.2}}} = \alpha^{r_1\alpha^{(r_{1.1}r_{1.1.1}r_{1.1.2})+(r_{1.2}r_{1.2.1}r_{1.2.2})}}$. As in the previous steps, each member, M_i, in cluster C_0 raises the received intermediate values, from its proceeding member in the root cluster, to the power of its private input and forwards the result to the next member in the cluster. The down-flow stage takes place when the last member, M_d, receives the last up-flow message and computes the group session key S_n. M_d then raises the intermediate values it has received to the power of its private key $\hat{r}_d$ ($\hat{r}_3$ in our example) and broadcasts the intermediate values to the other members, M_j ; $j \in [1, d-1]$, and their descendants. At the same time M_d broadcasts its cardinal value (of course before adding its secret exponent) to its descendants. For example, M_3 broadcasts $\alpha^{\hat{r}_1\hat{r}_2}$ to its descendants $\{M_{3.1}, M_{3.1.1}, M_{3.1.2}, M_{3.1.1.1}, M_{3.1.1.2}\}$, who have the value $\hat{r}_3$, broadcasts $\alpha^{\hat{r}_1\hat{r}_3}$ to M_2 and its descendants $\{M_{2.1}, M_{2.1.1}, M_{2.1.2}, M_{2.1.1.1}, M_{2.1.1.2}, M_{2.1.2.1}\}$, who have the value $\hat{r}_2$ and broadcasts $\alpha^{\hat{r}_2\hat{r}_3}$ to M_1 and its descendants $\{M_{1.1}, M_{1.2}$, $M_{1.1.1}, M_{1.1.2}, M_{1.2.1}, M_{1.2.2}\}$, who have the value $\hat{r}_1$. At this time all the group members are able to generate the group session key, $S_n = \alpha^{\hat{r}_1\hat{r}_2,\hat{r}_3}$. In practice, the generated session key is not used directly as a secret key for the subsequent private key encryption. Instead, we might hash the generated session key, S_n, to get a suitable length of the key. For example, if the generated session key, S_n, will be used as a symmetric key for AES-128, we can apply a cryptographic hash function like SHA-1 to S_n to yield 160 bits that might have better randomness properties than that of S_n. Now we can use, for example, the first 128 bit to get the 128 bits AES-128 key.

Algorithm 4.1 *Initial Key Generation Protocol (DTGKA)*

INPUT : l, d, C_p

OUTPUT: $S_n = \{\alpha^{r_1 r_2 \cdots r_d}\}$

1. For $i = 1$ *to* d

1.2 IF $ch(y) = 0$ *then* $M_y \leftarrow r_y \in G$

1.3 For each cluster, where C_{max} *is the cluster size*

1.3.1 For $y = 1$ *to* $(C_{max} - 1)$

1.3.1.1 $M_y \Rightarrow M_{y+1} : \{\alpha^{\frac{\Pi\{r_k\}}{r_k}}|_{k\in[1,y]}, \alpha^{r_1 \cdots r_y}\}$

1.3.2 $M_{y+1} \Rightarrow M_{L(y+1)} : \{\alpha^{\frac{\Pi\{r_k\}}{r_k}}|_{k\in[1,y+1]}, \alpha^{r_1 \cdots, r_{y+1}}\}$

1.3.3 $M_{L(y+1)} \Rightarrow \{M_{L(y+1)}\}_s : \{\alpha^{r_{L(y+1)}\frac{\Pi\{r_k\}}{r_k}}|_{k\in[1,y+1]}\}$

1.3.4 $r_{L(y+1)} \leftarrow \alpha^{r_{L(y+1)} r_1 r_2 \cdots r_{y+1}}$

2. IF $L(y+1) \neq 0$

2.1 Go to 1.3

2.2 Else

2.2.1 For $x = 1$ *to* $d - 1$

2.2.1.1 $M_x \Rightarrow M_{x+1} : \{\alpha^{\frac{\Pi\{r_k\}}{r_k}}|_{k\in[1,y]}, \alpha^{r_1 \cdots r_y}\}$

2.2.2 $M_d \Rightarrow \{M_z, \{M_z\}_s|_{z\in[1,d-1]}\} : \{\alpha^{\frac{r_1 r_2 \cdots r_d}{r_z}}\}$

2.2.3 $M_d \Rightarrow \{M_d\}_s : \{\alpha^{r_1 \cdots r_{d-1}}\}$

4.5.4 Security Analysis

In the previous protocol DTGDH, it is assumed that neighbouring nodes have authenticated each other, or at least verified each other's group membership, so that all messages are known to come from authentic members of the group. Consequently, it can be claimed that DTGKA protocol is secure against passive attacks, given the intractability of DHP or generally DLP, which is believed to be hard problem. This

assumption is realistic for different applications where the network is physically secure or when the communication channels are tamper-resistant.

On the other hand, it should be mentioned that DTGKA protocol is not secure against active attacks. Generally speaking, the design of key agreement protocol, where many parties are involved in a joint activity, allows dishonest adversaries to behave in arbitrary devious ways[1]. Thus, we need to ensure the identity of the participating parties. Most of the authenticated versions of the previous key agreement protocols used a signature for authentication, which is very costly (signature and verification). The weaknesses of using a pairwise long-term key for authentication has already been pointed out in [60]. In Section 4.6, an authenticated version (A-DTGKA) of DTGKA protocol is proposed, which is based on using identity based pairwise key and which addresses the issues and constraints mentioned above.

4.5.5 Performance Analysis

Beyond the security of the system, the complexity of the protocol (communication and computation) has always been an important issue when designing group key management systems. In ad hoc wireless networks, both communication costs and computation costs are important factors that should be taken into account when designing a secure protocol. On one hand, the mobile devices are often small and portable, and therefore, do not have much memory or computational power and they are probably not tamper-resistant. On the other hand, the connections in ad hoc networks are usually unreliable. Consequently, the number and size of messages should be reduced as much as possible, especially multi-hop messages. In DTGKA protocol, the total required computations are distributed among all the group

[1] A formal analysis for group key agreement protocols, including HGKM and CGKA, will be presented in Chapter 5.

members, which reduces the required computation power for each member. Although the number of required messages may be larger than that of the flat settings (GDH.2, Hypercube, Octopus, etc.), most of the manipulated messages are one hop messages, since the protocol confines the cluster members to the neighbours with one-hop distance. One-hop messages are much more reliable than multi-hop messages. Also it should be noted that the message size is much lower than that of the flat settings which reduces the required bandwidth. Finally, the efficiency of DTGKA protocol is based on the clustering scheme, which in most cases should provide better performance. As discussed in the previous chapter, the best efficiency can be achieved when the hierarchical structure becomes fully balanced, i.e., all the clusters have approximately equal size and the number of levels is not too large.

4.6 Authenticated DTGKA (A-DTGKA)

In this section, an authentication of DTGKA protocol will be presented which uses identity-based pairwise keys for entity authentication. In this scheme a master key, $\mathcal{S}$, can be stored distributively in order to minimize key escrow [27] with the trusted authorities (TAs). The expected group of nodes that would join the network will get their private keys from TAs, such that the private key of a user A can be computed as: $\mathcal{S}Q_A = \mathcal{S}_1 Q_A + \mathcal{S}_2 Q_A + \cdots + \mathcal{S}_k Q_A$, where the parameter $Q_A = \mathcal{H}(ID_A)$ and k is the number of the TAs in this scheme. Once every user has received the respective private key, every node can compute the pairwise shared secret key using the properties of Weil Pairing[2] (as proposed in [56]). For example user A and B can compute:

$$e(\mathcal{S}Q_A, Q_B) = e(Q_A, \mathcal{S}Q_B) = e(Q_A, Q_B)^{\mathcal{S}}$$

[2]For a brief description of Weil pairing properties, refer to Appendix B.

$$K_{AB} = e(Q_A, Q_B)^S$$

The identities of the users are assumed to be available publically. The shared secret key that is computed is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of some prime order q, where q is chosen to be very large number. To get a shared key we can apply a strong hash function to the resulting shared secret, i.e., $\overline{K}_{AB} = \mathcal{H}(k_{AB})$. Using this scheme, each pair of parties are able to have a shared secret key *without* any information being passed and without the risk of any attack (like man-in-the-middle) during information sharing as in the case of conventional two-party key agreement protocols. Finally, each member can authenticate itself to the others by merely encrypting, the transmitted messages using the pairwise shared key.

Below, we will describe the authentication scheme used for one cluster only, which can be applied to the entire group by a natural extension (Algorithm 4.2). In the following, $\mathrm{E}_K(m)$ means that the message m is encrypted with a key K. As an example, we consider have a cluster with four members, namely, A, B, C, D in which that D is the cluster leader. As usual, A picks a random value r_A and calculates α^{r_A} but instead of sending this value to the next member, it sends $\{\mathrm{E}_{K_{AB}}(\alpha^{r_A} \| ID_A) \| ID_A \| \mathrm{E}_{K_{AC}}(ID_A) \| \mathrm{E}_{K_{AD}}(ID_A)\}$, where ID_A is the identity of the member A, and $\|$ refers to a concatenation of two messages. Note that the member ID has been appended in plain-text to inform the recipient that this message is coming from A. Upon receiving this message, B checks the transmitter ID, uses the shared key with this member to decrypt the encrypted message. Finally B compares the encrypted ID with the plain-text one. If the check passes, A is authenticated by B, and since only B can decrypt the message, B is also authenticated by A. After the successful check, B proceeds with the second round of the protocol. B calculates $\alpha^{r_B}, \alpha^{r_A r_B}$ and forwards the rest of the message it has received with

its contributions to C, namely $\{\mathbf{E}_{K_{BC}}(\alpha^{r_A}\|\alpha^{r_B}\|\alpha^{r_A r_B}\|ID_B)\|ID_B\|K_{AC}(ID_A)\|$ $K_{BD}(ID_B)\|K_{AD}(ID_A)\}$. Note that the last two parts of the message are used to authenticate A and B to the cluster leader D. Upon receiving this message, C checks the identity of the transmitter, decrypts the first part of the message using the shared key with the transmitter where it is able to authenticate both A and B. C then calculates its contribution as mentioned in the DTGKA protocol and sends to D: $\{\mathbf{E}_{K_{CD}}(\alpha^{r_A r_B}\|\alpha^{r_A r_C}\|\alpha^{r_B r_C}\|\alpha^{r_A r_B r_C})\|ID_C \|\mathbf{E}_{K_{BD}}(ID_B)\|\mathbf{E}_{K_{AD}}(ID_A)\}$. As a last round, the cluster leader (D in our example) sends the following messages to its children:

$$D \rightarrow A : \mathbf{E}_{K_{AD}}(\alpha^{r_B r_C r_D}\|ID_D)\|ID_D$$

$$D \rightarrow B : \mathbf{E}_{K_{BD}}(\alpha^{r_A r_C r_D}\|ID_D)\|ID_D$$

$$D \rightarrow C : \mathbf{E}_{K_{CD}}(\alpha^{r_B r_A r_D}\|ID_D)\|ID_D$$

Upon receiving these messages, each member (A, B, C) checks the identity of the transmitter, decrypts the message and compares the transmitter identity in the cipher-text with the one in the plain-text. If the check passes, each member (A, B, C) can authenticate D. This allows for a pairwise authentication of the cluster leader D by its children. Also all the children can calculate the cluster session key, $S_c = \alpha^{r_A r_B r_C r_D}$. For the upper level clusters, the previously mentioned authenticated version will be repeated between all the cluster members. The only difference exists in the last round. In DTGKA protocol, the cluster leader broadcasts the final message to its children and their siblings. In A-DTGKA, the children decrypt the message and check the identity of the transmitter, re-encrypt the message using its cluster session key and resend it to its children. For example, (see Figure 4.1) in the last round (for this cluster) $M_{2.1}$ sends $\mathbf{E}_{K_{2.1,2.1.1}}(\alpha^{r_{2.1} r_{2.1.2}}\|ID_{2.1})\|ID_{2.1}$ to $M_{2.1.1}$ and its siblings ($M_{2.1.1.1}, M_{2.1.1.2}$). In this case $M_{2.1.1}$decrypts the message and re-encrypts it using its cluster session key, $K_{2.1.1}$, where, $K_{2.1.1} = \alpha^{r_{2.1.1} r_{2.1.1.1} r_{2.1.1.2}}$.

Algorithm 4.2 *Authenticated DTGKA (A-DTGKA)*

INPUT : $\alpha,\ p, \mathbf{E}_k,\ \mathbf{D}_k, \mathcal{H},\ s_l|_{l\in[1,k]}, ID_i|_{i\in[1,n]},\ \hat{e}(Q,Q)$

OUTPUT: $S_n = \{\alpha^{r_1 r_2 \ldots r_n}\}$

1. Initial setting: Each member, M_i, does the following:

1.1 $Q_i = \mathcal{H}(ID_i)$

1.2 $sQ_i = s_1 Q_i + s_2 Q_i + \ldots + s_k Q_i$

1.3 $K_{ij} = \hat{e}(sQ_i, Q_j)|_{j\in[1,n], j\neq i}$

2. Round i, $i \in [1, n-1]$

2.1 M_i selects $r_i \in \mathbb{Z}_p^*$

2.2 $M_i \xrightarrow{\mathrm{M} \parallel \mathbf{E}_{i(i+1)}(\mathrm{M} \parallel ID_i) \parallel ID_i \parallel \mathbf{E}_{ij}(ID_i)|_{j\in[1,n], j>i+1}\}} M_{i+1}$

Where $\mathrm{M} = \{\alpha^{\frac{r_1 \ldots r_i}{r_x}}|_{x\in[1,i]}, \alpha^{r_1,\ldots,r_i}\}$

2.3 M_{i+1} Checks ID_i and calculates $\mathbf{D}_{i(i+1)}(\mathbf{E}_{i(i+1)}(\mathrm{M} \parallel ID_i))$

2.4 M_{i+1} Compares ID_i with the encrypted ID_i

2.5 If the check passes go to 2.1

2.6 Else stop and send Error message.

3. Round n

3.1 $M_n \xrightarrow{\mathbf{E}_{in}(\{\alpha^{\frac{r_1 \ldots r_n}{r_i}}|_{i\in[1,n]}\} \parallel ID_n) \parallel ID_n} M_i$

3.1 M_i Checks ID_n and calculates $\mathbf{D}_{in}(\mathbf{E}_{in}(\{\alpha^{\frac{r_1 \ldots r_n}{r_i}}|_{i\in[1,n]}\} \parallel ID_n)$

3.2 M_i Compares ID_n with the encrypted ID_n

2.5 If the check passes calculate $S_n(M_i) = \{\alpha^{r_1 r_2 \ldots r_n}\}$

2.6 Else stop and send Error message.

This process continues up to the root cluster, as in DTGKA. Achieving authentication through private-key encryption makes A-DTGKA more efficient than the authenticated protocols which use a signature scheme, which is known to be very costly and more robust than hash-chains and time-synchronization authentication

schemes.

The protocol described above is an authenticated key agreement protocol which is based on hierarchical authentication. That is, each member authenticates its upper level members through its leader and authenticates its lower level members through its children. Although A-DTGKA does not provide a mutual authentication between all communicating parties, we believe that hierarchical authentication can be suitable in practical applications, like the hierarchical structure of trust which exists in military applications. Meanwhile, A-DTGKA can be modified to provide mutual authentication for all the communicating parties, but this will make the protocol more complicated and it can reduce protocol efficiency.

4.7 Summary

In this chapter, a proposed key agreement protocol that is efficient, robust and conformed the constraints and characteristics of ad hoc wireless networks was presented. The proposed protocol is a modification of the CGKA protocol presented in Chapter 3, where the distribution of clusters through the network is arbitrary and flexible to accommodate the dynamic nature of ad hoc wireless networks during the life of the session. An authenticated version of the DTGKA has also been presented. A-DTGKA is based on using Weil Pairing, where the pairwise shared key is generated without exchanging any messages. Moreover the mutual authentication comes through private-key encryption which is more efficient than using digital signatures. We did not address the question of the maintenance of the secret keys after membership changes, as they can be treated similar to those of CGKA protocol presented in the previous chapter.

Another point worth mentioning is that our proposal provides an easy solution to the mobility issue as if a member needs to move to join another cluster within the

group, he only needs to obtain the local parameters of this cluster and the cluster session key as well from the cluster leader encrypted by the global group session key. In this case, the member indexing should be changed according to its new cluster.

Chapter 5

Formal Security Analysis

In this chapter, a formal discussion of the security of the Group Diffie-Hellman (GDH) family of protocols is presented. Specifically, the formal security requirements for this family of protocols are discussed. A simple model for the analysis of this family of protocols is presented. Analysis of the Cliques Protocols is given as an example of the application of the proposed model. This analysis points out some weaknesses of the GDH protocols and possible attacks. Finally, we suggest modifications to these protocols that can avoid the discovered weaknesses.

5.1 Introduction

Authenticated two-party key agreement protocol allows two principals, A and B, communicating over a public network, and each holding a pair of public/private keys, to agree on a session key. Protocols designed to deal with this problem ensure A (B resp.) that no other principals aside from B (A resp.) can learn any information about this key or has the ability to calculate it. Some protocols, additionally, provide a key confirmation property which ensures A and B that their respective partner has

actually computed the shared secret value.

A natural extension to the above protocol is to consider a pool of principals agreeing on a session key. As it was discussed in Chapter 2, there have been several proposals that have been tried to extend the two-party Diffie-Hellman key exchange to n-party settings, such as the well-known Cliques cryptographic suite [5, 65, 66]. The authors of the Cliques suite have claimed that "*If a 2-party DH key is indistinguishable from a random value, then an n-party DH key is also indistinguishable from a random value*". So, most of their efforts were dedicated to the improvement of their protocols' performance in terms of bandwidth and computational requirements. Meanwhile, they exhibited only a *heuristic* analysis for the security of their protocols, which have been found to be flawed [59, 60]. These flaws are independent of the strengths or weaknesses of the used cryptographic algorithms, but are instead caused by the higher-level structure of the protocol itself where the nature of a session between a pool of principals is completely different than that between two principals.

One way to avoid many of these flaws is to make use of provable security methodology. The main goal of this approach is to provide a provably secure protocol, which up to formal representation assures the fulfilment of the security requirements of the protocol. It should be however noted that a particular care should be given to details and definitions of protocols using such analysis.

Pereira and Quisquater [59, 60] (PQ-model) have performed an analysis of the Cliques protocol by reducing the secrecy problem to a problem in linear algebra. They were able to demonstrate that a secret key can be compromised by a dishonest former member, even if it no longer belongs to the group. Although the authors follow a systematic way to analyze the protocol, their analysis is based on a wrong definition of the protocol as well as its intended requirements. As a consequence, most of their pinpointed attacks are unrealistic. In the rest of this chapter, a formal

model for the analysis of the family of key agreement protocols, based on group Diffie-Hellman (GDH) family, will be presented. The approach can be considered in the same line as that of the PQ–model, however, the reasoning and definitions of the known information and the intruder capabilities are completely different. Also, we should mention that the proposed approach is general and can be applied to any protocol that lies in the family under discussion. Using this approach, we are able to discover some practical attacks upon the Cliques family (as an example of GDH protocols). Also the approach leads to the main reasons behind these attacks and how they can be avoided.

5.2 Properties of Group Key Agreement Protocols

It has been pointed out [50] that it is difficult to design cryptographic protocols and guarantee their security, given the complexity of protocols and environment within which they work. Many problems with security protocols arise, not because the designed protocol did not satisfy its requirements, but because the requirements were not well understood in the first place. Before we can determine whether or not the intended key agreement protocol satisfies its requirements, it is necessary to determine carefully what those requirements are.

Let us first explain the key agreement protocol wherein we can specify its requirements: roughly speaking, the execution of any key agreement protocol allows a group of n users to contributively generate a shared group session key. This key should remain secret (i.e. no one outside the group should learn the key) even in the presence of an active intruder. Moreover, the protocol must provide key authentication and freshness. The main desired security properties for any key agreement protocol are as follows:

1. Implicit Key Authentication:

   Let $\mathcal{R}$ be an n-party key agreement protocol, $\mathcal{M}$the set of protocol parties, and S_n is a secret key jointly generated as a result of $\mathcal{R}$. We can say that $\mathcal{R}$ provides *Implicit Key Authentication* if each $M_i \in \mathcal{M}$ is assured that no party $M_q \notin \mathcal{M}$ can learn the generated key or be able to calculate that key using the manipulated intermediate messages, unless helped by a dishonest group member $M_j \in \mathcal{M}$. This property does not mean that the group members have any knowledge of the group key at the end of the protocol, nor that they agree on its value. Also, it does not imply for a group member that any member executed a session of the protocol.

2. Perfect Forward Security

   A protocol $\mathcal{R}$ provides *Perfect Forward Security*, if the compromise of any long-term secret keying material does not allow the compromise of the past session keys.

3. Resistance to Known-Key Attacks

   A protocol is said to be vulnerable to a known-key attack if compromise of past session keys allows either :1) a passive adversary to compromise keys of other sessions or :2) an active adversary to impersonate one of the authorized protocol parties.

4. Key Confirmation

   The Key Agreement Protocol is said to satisfy the property of key confirmation when at the end of the protocol, each member of the group is assured that the other member(s) actually has possession of a particular secret key. Technically speaking, when both implicit key authentication and key confirmation

are held by a Key Agreement Protocol, the property of Explicit Key Authentication is obtained, i,e., when one party has assurance that only a particularly identified other party actually has knowledge or possession of the generated key.

The first glance at these requirements gives an impression that these requirements are very similar to those requirements of a two-party key exchange protocol. But when we look at these requirements more closely, we can see some important differences. These differences comes from the fact that the notion of "session", which is so important to 2-party protocols, does not exist, at least in the same sense, for group protocols. For example, in a pairwise protocol, a session is determined by two communicating parties and if one of the two members leaves, the whole session will be ended.

As the first requirement states, keys should not be learned by anyone other than the communicating parties, and a key should be unique to a certain session. However, group protocols usually do not have such a notion of individual sessions strongly tied to parties and keys. Instead, the paradigm is of a group which parties may enter and leave a group at any time. However, the incoming party should have no access to the old keys, and the leaving party should not have access to new keys. Moreover, keys may be updated for other reasons, like periodical key refresh, that have nothing to do with the group composition. Dealing with this situation is a little bit harder than that of two-party settings, since to fulfil this property we have to determine clearly what is the *view* of the present session i.e., who is actually participating in the present session. Moreover, the deleted or new members should be considered as outsiders for the present session.

The second property was traditionally used in a two-party settings, where no messages were manipulated in the past. This is natural since the two parties cannot

exchange messages before agreeing on the used key. However things are slightly different in a group setting where the manipulation of a message can alter the view of the key of one particular user, while all the other group members are computing the same key. This manipulation does not prevent these last users from communicating.

The third property has no meaning in two-party setting, since in practice, the used key is per session only. The session key can be changed at the beginning of each session to guarantee the freshness of the key. This procedure is very costly in the case of group communication, which suffers from a large group size and fast membership changes.

The last property is very critical in the case of group communication, similar to that of two-party communication, to prevent man-in-the-middle attack. Although this property requires one more round to achieve, it is more costly in the case of group communication than that of two-party communication. But we have to be careful with this property, since the common solution of exchanging a fixed message between users encrypted with the agreed session key is insecure. Partial information about the session key can be easily obtained from this exchange.

We should reemphasize that the previous requirements should have the ability to be fulfilled under very adverse conditions. In general, it is assumed that the intruder has complete control of all communication channels (which is practical, since most communication passes through insecure networks like the Internet), and thus can read all traffic, destroy or alter traffic, and generate traffic of its own. Also, we should not suppose or consider that the computational power of the intruder is lower than that of the most powerful member. However, it is not necessary to assume the existence of multiple attackers, because together they have no more abilities than the single powerful attacker. It is also usually assumed that some users are cooperating with the intruder, and thus that intruder will be able to perform

operations that are also available to honest users (we will not consider this attack in our discussion). Protocols which consider this situation are referred to in the literature as "Byzantine Agreement" [57].

5.3 Survey of Provable Security Techniques

From the previous discussion, it is clear that the design of security protocols, especially those used to secure distributed groups in a very hostile environment full of active attacks, is a particularly hard task. In this section, a brief discussion of the two approaches used for analysis, and proof of the security of cryptographic protocols is presented (for more discussion see [2]):

The Computational Approach: This approach is based on the framework of computational complexity theory. Where keys, plaintexts and ciphertexts are just strings of bits; an encryption function is defined as an algorithm and the adversary is essentially a Turing machine with enormous capabilities. The proof of security of these protocols is defined through the success probability of the adversary, which is based on its capability. With regards to proving the protocols concerned with exchanging a session key between players, the first model was initiated by Bellare and Rogaway who modelled two-party and three-party key distribution [10, 12]. In this model, the instances of a player are modelled via oracles and the capability of the adversary are modelled through queries to these oracles. This model was extended to the authenticated two-party Diffie-Hellman key exchange by Blake-Wilson *et al.* [14] and extended further by Bellare *et al.* to deal with password-based authentication 2-party key exchange [11]. More recently, Bresson *et al.* further extended these methods for the analysis of authenticated group key agreement protocols [16, 17]. Although a security proof using a computational approach is of some value, it is still only a heuristic proof. In particular, these types of proofs do

not rule out the possibility of breaking the scheme without breaking the underlying intractability assumption. Nor do they even rule out the possibility of breaking the scheme without finding some kind of weakness in the cryptographic functions used, such as the one shown by Cramer and Shoup [25] in the case of hash function.

The Formal Approach: The main purpose of a formal model is to examine the ways that malicious attackers can interfer with a security protocol and cause it to malfunction. These methods suggest ways to defeat theses attacks. A number of methods were developed during the last decade for the formal (or logical) analysis of security protocols. Many of them are based on *state exploration*, that is they try to find as many paths through the protocol as possible, in the hope that, if there is an error, it will be discovered. But, since the search space offered by a cryptographic protocol is infinite (specially in the case of secure distributed communication), this search alone cannot guarantee security if no attack is found. Other approaches are based on the use of *Logics* [68]. They allow proofs to be obtained for arbitrary size configuration, but they often require particularly error-prone formalization steps and do not provide the same support in pinpointing problems as the direct generation of counter examples. In [70], P. van Oorschot extends and refines BAN logics [20] to examine and analyze some of Authenticated Key Agreement protocols. He formalizes 6-common goals and checks whether these protocols, under analysis, fulfil these goals or not by idealizing these protocols and checking each protocol through the proof of some lemmas. Although the formal model used in analyzing these protocols have proved that the protocols have fulfilled the required goals, it fails to discover possible active attacks and so it fails to manifest how to modify these protocols to avoid these possible active attacks. Recently C. Meadows [49] has performed the analysis of the A-GDH.2 protocol [5], adapting her NRL protocol analyzer [48] by extending the power and scope of its theorem-proving capabilities. At the same time (and independently), Pereira and Quisquater [59, 60]

introduce another formal model (PQ-model) to analyze protocol suites extending the Diffie-Hellman key exchange scheme to a group setting and was presented in the context of the Cliques project [66].

5.4 New Approach to the Analysis of GDH Protocols

In this section, we introduce a new approach for modelling and analysis of GDH, which can be extended to the proposed protocols (HGDH, CGDH). The proposed approach is based on the facts learned from the previous attempts in analyzing the key agreement protocols:

1. The security of the 2-party DH key exchange protocol does not imply the security of n-party DH protocols. In other words, the confidence of the security of the 2-party DH-protocol should not be directly extended to the n-party DH protocol due to the difference of the requirement definitions and consequences which were mentioned in section 5.2.

2. The concurrent conduction of the group key agreement protocol adds additional consideration to security requirements. For example, we have to consider the interleaving attacks, where the attacker uses selective combination of information from ongoing protocol executions [52].

3. The intruder capabilities can be mapped to the given (publicly transmitted or broadcasted) data and also to the available services which enable intruders to attack the protocol. The last part should consider active capabilities of the intruder, where it can inject a new message, reorder existing messages, force an honest member to produce a certain message, and duplicate or delete certain messages. In summary, we have to consider that the intruder has complete control on the transmitted messages.

Naturally, cryptographic protocols cannot be stronger than the cryptographic primitives they employ. Consequently, security protocols can also contain flaws that are not a consequence of some cryptanalytic flaws in the used algorithms, but are instead caused by the higher-level structure of the protocol itself. Thus, such flaws are more or less present despite the choice of the cryptographic functions used to implement the protocol. In this model, it is assumed that the individual underlying cryptographic primitives, DDH in the protocol, to be analyzed, are secure and are not vulnerable to attacks. So we restrict our analysis to attacks and flaws in the protocols themselves.

The model can be summarized in the following steps:

1. *Definition of the Protocol*

   The first step is to describe the required steps to conduct the protocol and, carefully, describe the role of each user. Moreover, the secret data and the goal(s) of the protocol must be specified. Concurrent steps must be differentiated from serial ones. Each protocol has certain assumptions that should be followed correctly. Also the requirements which the protocol is intended to satisfy should be specified. As has been mentioned earlier, it is impossible to design a protocol that fulfills all the security requirements for all applications. Consequently, the analyst should limit the analysis to certain specified requirements.

2. *Passive Intruder Capabilities*

   By passive capabilities, the model addresses the available data to the intruder. The publicly transmitted and/or broadcasted data and the publicly stored data must be specified. This data is a result of one or more steps performed by an honest member through the conduction of the protocol and it will be referred

to as Honest Services (S_h)[1].

3. *Active Intruder Capabilities*

   The active capabilities (injection, reordering, deleting, etc.,) of the intruder should be clearly specified. These capabilities will be referred to as Intruder Services S_i. The services, where the intruder forces (or tricks) an honest member (or simultaneously more than one member) to perform a certain service or to reveal some information such that the intruder can obtain some data he could not otherwise obtain, will be known as Forced Services (S_f).

4. *Protocol Analysis*

   At this step we need to check if the intended protocol requirements have been fulfilled or not and to verify if any secret has been divulged through the conduction of the protocol or not. The check is performed through a certain linear system of equations where the variables are known data and a specific formulation of the available intruder services (S_h, S_i, S_f) and the unknown values are all the secrets in the protocol. Each requirement can be fulfilled if a certain secret is not divulged through the conduction of the protocol, so each requirement will be analyzed separately, in other words each secret will be checked separately.

5. *Security Proof*

   This system of equations can now be checked, if it is inconsistent (there is no solution), then we can prove that the corresponding requirement is verified. If this is not the case, i.e., there is a solution to these equations, then a certain attack can be constructed, even if we could not build this attack. In the later

[1]The word '*honest*' means that a member proceeds exactly according to the protocol being verified and does not divert from the protocol specifications at any moment.

case we have to go back to step 1 and try to modify the protocol to defeat this attack. In the end, if the system of equations of all the requirements (protocol secrets) are inconsistent, we can say that the protocol is secure (can fulfill the specified requirements) in our model and under ceratin assumptions.

5.5 Modelling of GDH Protocols

In this section, a formal model of the GDH family is presented. By this family, we mean protocols that use a modular exponentiation as primitives to generate a group secret key. These protocols do not use any other cryptographic primitives like signature/verification or/and hash functions through the calculation of the secret key. These protocols use abelian cyclic groups $\mathbb{G}$ of prime order p as their underlying field. The security of these protocols is reduced to the intractability of the DLP and its equivalent DHP.

Definition 5.1 *Let $\mathbb{G}$ be a finite cyclic group of order p. Let α be a generator of $\mathbb{G}$, and let $\beta \in \mathbb{G}$. The discrete logarithm problem of β to the base α, denoted $\log_\alpha \beta$, is the unique integer $x; 0 \leq x \leq p-1$, such that $\beta = \alpha^x$.*

5.5.1 Protocol Description

A description of GDH protocols is first presented. Generally, these protocols consist of two steps: the first step considers gathering the contributions from all participants. These contributions, $r_i \in \mathbb{G}\ ; i = 1 \ldots n$ (where n is the total number of participants) should be collected in secret.

For example, using a multiplicative group, the secret r_i is collected in the form of $\alpha^{r_i}\ mod\ p$, where α is the generator of the group. This step can be done generally in two different ways. The first one (unicast) as simple as a member updating the re-

ceived message from its previous member by adding its public share and forwarding this piece of information to its neighbour. In the second way (broadcasting), group members, or subset of the group members, simultaneously broadcast their contribution to the other group members. These contributions may be based only on the members' public shares or based on their current information. The second step is the construction of the common secret as a function of members contributions $\mathcal{F}(r_1, r_2, \ldots, r_n)$. In the authenticated version of these protocols a long-term pairwise shared key, K_{ij}, is used in different ways to authenticate participant M_i to M_j. It can be checked that the message sent during the execution of the protocol can be expressed as α raised to the power of a product of the subset of secret contributions of all participants. We can now distinguish between three types of variables each participant, M_i, has to deal with during the execution of the protocol:

- Secret contribution ($r_i \in \mathbb{Z}_P^*$).
- Pairwise shared long-term key with participant M_j (K_{ij}).
- $\alpha^{\frac{\Pi r_i}{r_i}}|_i \in [1,n]$ or $\alpha^{\frac{\Pi r_i}{r_i} K_{ij}}|_i \in [1,n]$

Note that the first and second variables should be kept secret all the time, while the last variables are transmitted in plain form during the execution of the protocol. We can consider r_i as a secret share of participant M_i, while α^{r_i} is its public share. As mentioned before, there are different ways to collect the participants' contributions and also there are a different functions $\mathcal{F}$ to construct the common secret. Since the aim of the intruder, in this family of protocols, is to get as much information which enables it to calculate the session key, we will only consider the multiple use of random variables and/or the long term-key in our model. From the above, we can claim that this family of protocols manipulate two types of variables:

- Secret variables $\mathbf{S} = \{r_I, r_i,\ K_{ij}|_{i,j\in[1,n],i\neq j}\}$

- Public variables $\mathbf{P} = \{\frac{\prod r_i}{r_i}\}$ and $\{\frac{\prod r_i K_{ij}}{r_i}|_{i,j\in[1,n],i\neq j}\}$

The term r_I in the first set is the intruder secret share, which can be inserted at any stage of the protocol. This term will be considered, by the honest members, as an honest member's secret share. Note that both secret variables s can be manipulated by itself s or its inversion s^{-1}. The generated secret key at the end of the protocol is calculated as $S_n = \alpha^{r_1 r_2 \cdots r_n}$. Beside these manipulated variables, the group members use other variables most probably in the last round, these variables enable each member to calculate its view of the generated session key. Since these secrets are required to calculate the session key and they are the main target of the intruder, these variables will be called Required Secrets $\mathbf{R}_S$. In protocols where the group members are distributed in a hierarchial structure, a low level generated secret can be used as a secret share of upper level members. For example, if the group members are distributed in different levels, where at a certain level the number of members is d, the generated secret will be $S_d = \alpha^{r_1 r_2, \cdots, r_d}$. This generated secret will be used by the upper level member as its secret share and its public share will be $\alpha^{\alpha^{r_1 r_2, \cdots, r_d}}$. This process can be repeated according the group structure. In these protocols, one assumes that there is a bijective mapping from the generated secret in one round, which is an element in $\mathbb{G}$, to the random element used as a secret in the subsequent round, which is an element of $\mathbb{Z}_q^*$.

The last step of the protocol description is to formulate the requirements which the protocol intends to fulfill. We consider in our analysis that the analyzed protocol is an authenticated one, which means that it should fulfill all the previously mentioned requirements. During the analysis, we will check each requirement in turn.

5.5.2 Intruder Capabilities

As mentioned earlier, we consider the existence of an active intruder who controls all the communications between the participants. So the intruder can perform or access services (S_h, S_i S_f). If we consider that $\mathbf{S}_I$ and $\mathbf{P}_I$ are the subset of $\mathbf{S}$ and $\mathbf{P}$ that are known to the intruder, we can then discuss the available services as follows:

1. $S_h \Leftrightarrow \mathbf{P}_I = \mathbf{P}$
2. If $s \in \mathbf{S}_I$ and $p \in \mathbf{P}_I$, then $S_i(M_I) \Rightarrow sp \in \mathbf{P}_I$
3. If $s \in \mathbf{S}_I$, then $S_f(M_x) \Rightarrow sxp \in \mathbf{P}_I$

The first intruder capability, S_h, model the passive intruder, who is able to intercept all the public variables result from correct execution of the protocol by honest members. While the second service S_i represents the intruder capability, who exponentiates an element of $\mathbf{P}_I$ with a known element of $\mathbf{S}_I$. The service S_f models capabilities of an active intruder, where $p \in \mathbf{P}$, represents the result from the computation executed by the preceding member. If the intruder deletes the preceding member results and inserts a totally different value s, then $p = 1$. Since honest members have no means to check whether the messages it has received is a correct message or not, we can consider that this assumption is correct. The only condition to accept this assumption is that the inserted values are elements of the underlying algebraic group $\mathbb{G}$.

5.5.3 Security Check

After protocol description and modelling of the intruder capabilities, we model the known variables and the intruder capabilities as a set of linear equations and check whether these set of equations has a consistent solution or not. Finding a solution to

this set of equations implies that the intruder is able to construct a practical attack to the protocol, which leads to the calculation of secrets, and finally enables the intruder to calculate the session key. The building of this set of equations is based on the secrets that the intruder wants to access and it is controlled by the protocol assumptions.

5.6 Analysis of Cliques Suite

In this section, the proposed model will be applied to analyze GDH.2, which is one of the protocols presented in the Cliques suite. During the analysis, the validity of each requirement is checked in turn.

5.6.1 Implicit Key Authentication

In this discussion, we will consider two cases. In the first case, the intruder is not a member of any group. We can consider this case, when a new member joins the group and we need to check whether this new member can calculate the current session key or not. In the second case, we consider the situation where the intruder was a legitimate member of some previous session.

The Intruder is a Non-Legitimate Member

In this case, we have to discuss the existence of an intruder in all possible sessions, i.e. during initial key generation, member join and member leave and check whether the intruder is able to access any secret information which enables the intruder to calculate the generated secret or not. As described in Section 4.5, the first step in the analysis will be the protocol description. We consider a session of the protocol with n participants. The intruder has no information related to the present session,

i.e., the sets S_I, P_I are empty. The intruder may have access to some public information from the previous sessions. But this information will not help him in the present session. To get any valuable information from the manipulated message, the intruder has to solve the DLP, which is believed to be a hard problem. As the protocol assumes, the public and secret values can be summarized as follows:

$$\begin{aligned} \mathrm{S} &= \{r_1, r_2, ..., r_{n-1}, r_n K_{1n}, ..., r_n K_{(n-1)n}\} \\ \mathrm{S}_I &= \phi \\ \mathrm{R}_S &= \{r_i K_{in}^{-1}|_{i\in[1,n-1]}, r_n\} \end{aligned}$$

As the model describes, we need now to express the system of linear equations, based on the given information, to check if the given information enables the intruder to get access to any of the elements in the set R_S. Each element in R_S has to be checked separately. For example, if we need to check the secrecy of $r_2 K_{2n}$. The set of linear equations can be represented as follows:

$$\begin{aligned} & r_i = 0 && i \in [1, n-1], i \neq 2 \\ & r_2 = 1 \\ & r_n + \textstyle\sum_{i=1}^{n-1} r_n K_{in} = 0 && i \in [1, n-1], i \neq 2 \\ & r_n K_{2n} = -1 \\ & r_n K_{in} = 0 && i \in [1, n-1], i \neq 2 \end{aligned}$$

It can be observed that this set of equations does not provide any new information other than it assumes a balance of r_2 and $r_n K_{2n}$. Hence, we can say that the intended element of R_S can not be obtained from the available information, and hence the session key is kept secret. Application of the model to any element in R_S is straightforward and gives the same result. We can say then that the intended protocol satisfies the implicit key authentication when the member is not a legitimate member of any session.

The Intruder was a Legitimate Member

Now, we consider the case where the intruder was a legitimate member of some previous sessions, then it is evicted from the group. The case where the intruder was a legitimate member of a very old session (i.e. there are many membership changes and consequently there have been much of key regeneration processing separating the present session from the session where the intruder was a legitimate member) can be considered as the previous case.

As a simple scenario, we assume a first session of the protocol in which M_1, M_2, M_I, M_3 are the intended participants (M_3 being the group controller), and in the second session, the member M_I has been evicted. In the second session (as it was explained in [5]), the group controller (M_3) picks another secret share ($\acute{r}_3$) and raises the stored intermediate messages with $\acute{r}_3$ and the long-term key ($K_{i3};\ i = 1, 2$) and sends the result to the remaining group members (M_1, M_2). So, the sets of interest become:

$$\begin{aligned} \mathbf{S} &= \{r_1, r_2, r_I, r_3K_{13}, r_3K_{23}, r_3K_{I3}, \acute{r}_3K_{13}, \acute{r}_3K_{23}\} \\ \mathbf{S_I} &= \{r_I, K_{I3}\} \\ \mathbf{R}_S &= \{r_1K_{13}^{-1}, r_2K_{23}^{-1}, \acute{r}_3\} \end{aligned}$$

Removing the variables known by the intruder will helps in the analysis and will not affect the result. The remaining set of interested variables will be as follows:

$$\begin{aligned} \mathbf{S} &= \{r_1, r_2, r_3K_{13}, r_3K_{23}, r_3, \acute{r}_3K_{13}, \acute{r}_3K_{23}\} \\ \mathbf{R}_S &= \{\, r_1K_{13}^{-1}, r_2K_{23}^{-1}, \acute{r}_3\} \end{aligned}$$

Now we check the security of the elements of the set R_S^1. We build the linear systems, one for each secret. We will give the example of the verification of the secrecy of $r_2K_{23}^{-1}$. The linear system corresponding to this secret can be written as

follows:

$$\begin{aligned} r_1 &= 0 \\ r_2 &= 1 \\ r_3K_{13} + r_3K_{23} + r_3 &= 0 \\ r'_3K_{13} + r'_3K_{23} &= 0 \\ r_3K_{13} + r'_3K_{13} &= 0 \\ r_3K_{23} + r'_3K_{23} &= -1 \end{aligned}$$

There are two different solutions to this linear system:

1. $r_2 = r_3 = 1,\ r_3K_{23} = -1,\ r_1 = r_3K_{13} = r'_3K_{13} = r'_3K_{23} = 0.$

2. $r_2 = r_3 = r'_3K_{13} = 1,\ r_3K_{13} = r'_3K_{23} = -1,\ r_1 = r_3K_{23} = 0.$

To build attacks related to these two solutions, we have two scenarios and both are based on the intruder having its contribution in the session key. This share will enable him to get access to the session key of the next session which he is already evicted from. The first scenario runs as follows:

In the first session, the intruder does not do anything, i.e. the protocol steps run as usual and the generated session key for all members will be the same $S_n(M_i) = \alpha^{r_1r_2r_Ir_3}; i = 1, 2, I, 3$. In the normal protocol processing of the second session, the group leader M_3 picks another random secret $\hat{r}_3$ and raises the values he has received in the previous session and sends $\alpha^{r_2r_Ir_3\hat{r}_3K_{13}}$ to M_1 and $\alpha^{r_1r_Ir_3\hat{r}_3K_{23}}$ to M_2, where the remaining group members calculate the key $S_n(M_i) = \alpha^{r_1r_2r_I\hat{r}_3r_3}; i = 1, 2, 3$. If the intruder would like to attack M_2 for example, he just replaces the message intended for M_2 ($\alpha^{r_1r_Ir_3\hat{r}_3K_{23}}$) by another message which enables the intruder to get access to the session key, $S_n(M_2)$. For example, the intruder inserts the message $\alpha^{r_1r_3K_{23}}$. This message can be obtained by the intruder if he removed his share from the message sent to M_2 in the first session. By the end of the second session, $S_n(M_2) = \alpha^{r_1r_2r_3}$ is accessible by the intruder. While $S_n(M_1, M_3) = \alpha^{r_1r_2r_I\hat{r}_3r_3}$.

This attack can be applied simultaneously to M_1 if the intruder does the same with M_1 where it will be forced to calculate $S_n(M_1) = \alpha^{r_1 r_2 r_3}$ as a session key. By applying this attack to all group members, except the group controller, the intruder can get access to the session key calculated by all the group members (except the group controller) in a session from which he is normally excluded.

In the second scenario, the intruder does the reverse, i.e., he removes his share in the first session from the message sent to the fooled member (M_2 for example), where M_2 calculates $S_n(M_2) = \alpha^{r_1 r_2 r_3}$ in the first session and calculates $S_n(M_2) = \alpha^{r_1 r_2 r_I r_3}$ in the second session. Both values are accessible by the intruder. Also this attack can be applied simultaneously upon more than one member (it can be applied upon all the group members except the group controller).

The main idea behind this type of attack is that the intruder has its share in the messages manipulated in the first session (and in the generated session key), which enable him to play around changing the messages sent to the rest of the group members although he is evicted from the next sessions. We have to mention that changing the intruder's place in the ring sequence will not change anything in the capability of the intruder to attack the protocol.

Recalling the definition of implicit key authentication, the intruder is able to calculate the view of the generated session key, of one or more group members, from the manipulated intermediate messages. To conclude, implicit authentication property seems to be problematic in Cliques protocol for the case where the intruder was a legitimate member of some sessions.

We should mention that this attack is not a full attack to the whole group, since the intruder can intercept the messages sent by the fooled member(s) only. Moreover, this attack can be easily detected by some sort of management. For example, if the the session is interactive, the rest of the group members will discover the absence of this member. Simply because neither the intruder nor the fooled member(s)

have no access to the session key.

In comparison, the PQ-Model analysis [59, 60] of the implicit key authentication of for Cliques protocol, considers that the whole protocol is repeated when any member leaves the group, which is not the case. As mentioned in the case of a member departure, the last member (group controller) who is supposed to keep the last messages he has received from the last process (initial key generation or membership changes), changes his secret share and broadcasts the new intermediate values to the rest of the group members raised by the long-term secret key which is shared between group controller and the intended member, in such a way that it prevents the deleted member from being able to calculate the new session key. The assumption made in the PQ-model gives more opportunities to the intruder and eases his mission by adding some extra messages and rounds which do not exist in practice. Consequently, the attacks discovered by the PQ-model are not practical attacks.

5.6.2 Perfect Forward Secrecy

In the study of this property, which states that the compromise of any long-term key should not compromise the session key, we suppose that the intruder knows one (or more) long-term key related to one (or more) authorized member(s). We can build the attack simply by making the active intruder delete the intermediate value corresponding to the fooled member and use instead the cardinal value. In this case, the intruder can get the session key simply by adding the inverse of the fooled member's long-term key. For example, consider a session with 4 members and consider that the intruder knows the value of the long-term key corresponding to the member M_1 (K_{14}), the last message received by the fourth member will be $\alpha^{r_2r_3}$, $\alpha^{r_1r_3}$, $\alpha^{r_1r_2}$, $\alpha^{r_1r_2r_3}$; the cardinal value here is $\alpha^{r_1r_2r_3}$. In this case, the group controller (M_4) broadcasts the following message $\alpha^{r_2r_3r_4K_{14}}$, $\alpha^{r_1r_3r_4K_{24}}$, $\alpha^{r_1r_2r_4K_{34}}$. In this case, all

the members are able to calculate the session key, $\alpha^{r_1r_2r_3r_4}$. If the intruder, for example, replaces $\alpha^{r_2r_3}$ (this intermediate value corresponding to M_1) by the cardinal value, the group controller will receive this message $\alpha^{r_1r_2r_3}, \alpha^{r_1r_3}, \alpha^{r_1r_2}, \alpha^{r_1r_2r_3}$ and broadcasts $\alpha^{r_1r_2r_3r_4K_{14}}, \alpha^{r_1r_3r_4K_{24}}, \alpha^{r_1r_2r_4K_{34}}$. We can see that the session key is publicly broadcasted so any intruder who knows K_{14} can get the session key. Meanwhile, the intruder can share with member M_1 another key, for example, he can pick up $\alpha^{r_2r_3}$ and add any exponent r_I and send the message $\alpha^{r_2r_3r_IK_{14}}$ to M_1, in this case M_1 will share with the intruder the key $\alpha^{r_1r_2r_3r_I}$. This value the intruder is able to calculate by raising the cardinal value $\alpha^{r_1r_2r_3}$ by its share r_I.We can see that this property seems to be problematic and needs some effort to overcome.

It should be mentioned that the classification of the Perfect Forward Secrecy into two categories (Individual Forward Security and Complete Forward Security), which has been assumed in PQ-model, has no meaning in practice. The Intruder can employ his attack and masquerade as any member (or a set of members simultaneously) and at the same time he can get the group session key as will be shown. Moreover, our proposed attacks is simpler and practical and allows the intruder to share another key with the fooled member. In this case the intruder can intercept all the communication between group members. Moreover, the intruder can fool the other group members by convincing them that the fooled member is still alive if a key conformation property is provided by the protocol. Also the difference in the session key will not be noticed. We can see that the perfect forward secrecy property is problematic in the Cliques protocol especially with the existence of an active intruder. The Cliques protocol is seem to be only secure against passive attacks as discussed in [5].

5.6.3 Resistance to Known-Keys Attacks

To check if GDH-2 protocol satisfies this property or not, we anew consider the two cases. Namely, when the intruder was a legitimate member of some sessions and when the intruder is excluded from all sessions.

The latter case can be treated in the same manner as the analysis of the implicit key authentication when the intruder is excluded from all sessions. The difference in this case is that the intruder has access to the previous session key $\alpha^{r_1 r_2 \cdots r_n}$ where n is the group size. Since the intruder has no access to any secret in the present session S_I is still an empty set. Consequently, the analysis of this property is the same as that of the implicit key authentication case. Hence, we can say that the intruder is not able to get access to any element of R_S from the available information, and hence the session key is kept secret. We can say then that the intended protocol satisfies the resistance to the known-key property when the member is not a legitimate member of any session.

In the other case, where the intruder was a legitimate member of some previous sessions, he also would have access to some intermediate messages manipulated during the session when he was a legitimate user. In this case, the analysis should consider the existence of some extra information. This also will be the same situation when we consider the implicit key authentication when the intruder is a legitimate member in some previous sessions.

To conclude, the known-key attacks are not available in the case of the Cliques protocol. The intruder who knows a past session key(s) is neither able to compromise keys of other sessions nor impersonate one of the group members. It should be mentioned that the first part of this property (compromise of past session keys allows a passive adversary to compromise keys of other sessions) has been discussed in [5] and the authors have shown that it is not problematic.

5.7 Discussion

During the analysis to GDH.2, which is the most important protocol in the Cliques suite, some serious weaknesses have been discovered. These weaknesses are due to some facts which we have learned from this analysis:

- Most of these attacks come from the fact that the properties of group key agreement protocols are not trivial extensions of the ones of two-party protocols as discussed in Section 5.2. The discovery of these weaknesses emphasizes the need for using a formal definition of the protocol requirements. These requirements should be related to the nature of the underlying application.

- Using a long-term key in its plain form for authentication is a problematic approach. We have shown that getting access to the long-term key of one or more members enables the intruder to get access to the generated session key. The compromise of the long-term key is a common and practical problem due to the frequent and multiple use of this long-term key. Since authentication is a very crucial step, and should be accompanied with any key agreement protocol, we have to use a long-term key in such a way that it helps in entity authentication but at the same time it makes it difficult for any intruder to get access to the long-term key despite its frequent use. One approach, given in the next chapter, is to couple the long-term key with the short term one, r_i, in such a way that it is very difficult for any intruder to get access to the long-term key in its plain form.

- Although the key confirmation property is mentioned in most of the literature as a security requirement for key agreement protocols, it is rarely used in practice. This may be because applying key confirmation requires extra costs.

In our analysis, we have shown that key confirmation is a crucial property and most of the discovered attacks can be defeated if the protocol supports key confirmation property in a proper way.

5.8 Summary

Naturally, cryptographic protocols cannot be stronger than the cryptographic algorithms they employ. Cryptographic protocols can contains flaws that are independent of the strengths or weaknesses of the cryptographic algorithms used, but are instead caused by the higher-level structure of the protocol itself. If systematic means of assuring correctness are not used, these flaws may not be discovered until significant damage has been done.

However, misunderstanding of the protocol's requirements and characteristics may lead to a wrong judgment. Consequently, the analysis of the protocol may declare unpractical attacks while real ones still exist. This may cause protocol designers to modify their protocol where some extra cautions can be added, which may affect protocol performance. More seriously, these added steps may allow other types of attacks to appear, which can be considered as a reverse feedback.

In this chapter, we showed that some published attacks upon the Cliques protocol do not seem very useful in practice. At the same time, the proposed analysis pinpointed other types of attacks. Our analysis clearly showed that using a long-term key for authentication, as it is commonly used in practice, can be problematic. In the following chapter, several solutions to provide authentication in group key agreement protocols will be presented.

Chapter 6

Authenticated Group Key Agreement

In this chapter, we will propose new authentication technique to be incorporated within the proposed group key agreement protocols. Specifically, different authenticated group key agreement protocols will be presented. The first two protocols, GMTI and GMQV, use the set of points on an elliptic curve as its underlying algebraic group and are suitable for static small size networks with a fixed topology. The third one, ABD, uses a multiplicative group as their underlying algebraic group and it is suitable for larger dynamical networks with any topology such as ad hoc wireless networks.

6.1 Introduction

The main concern of this thesis so far has been the efficiency of group key agreement protocols. In the previous chapter, we gave a formal analysis of the existing and proposed protocols. This analysis clearly indicated serious flaws due to the improper use of long-term key for authentication. In this chapter, we will propose several solutions for authentication in group key agreement protocols, in particular

GMTI, GMQV and ABD protocols. Advantages of each and their limitations will also be discussed.

There are many solutions presented in the literature that provide authentication with key agreement protocols. For example, some apply digital signatures, encryption and hashing functions to provide authentication. One important factor that should be considered, using these approaches or any other primitives for authentication, is the costs involved. For example, if digital signatures are used for authentication, the authenticated key agreement protocol requires two types of algorithms: the Diffie-Hellman for key agreement scheme and a public key signature scheme (such as RSA) for authentication, which burdens the computational cost of the implementation. An example of this approach has been recently proposed in [40]. One may try to address this issue by using the same basic function for both schemes, but this still requires extra computations [62]. Our approach is to provide authentication within group key agreement protocols by modifying the content of the messages. In other words, the proposed authentication technique uses the same primitives used and is done within the group key agreement protocols steps. This in contrary to some belief, such as the one stated by O. Pereira claims, in his Ph.D. thesis [58] that *"it is not possible to write a protocol based on the same design assumptions than the A-GDH ones and provide implicit key authentication"*. In the remaining of this chapter, we will present different proposals for authenticated group key agreement protocols using the same primitives.

6.2 Initial Setting

All protocols described in the previous chapters have been described in the setting of a multiplicative group. However, they can be easily modified to work in any finite abelian group in which the discrete logarithm problem appears to be intractable.

Suitable choices include the multiplicative group of a finite field, subgroups of $\mathbb{Z}_n^*$ where n is a composite integer, subgroups of $\mathbb{Z}_p^*$ of prime order q, and the set of points on an elliptic curve defined over a finite field. Elliptic curves are often preferred because they appear to offer equivalent security to the other groups but with smaller key sizes and faster computation times.

The common parameters, for the following two protocols, GMTI and GMQV, consist of a suitably chosen elliptic curve E, over a finite field $\mathbb{F}_q$, of characteristic p and a base point P. The set of points on this curve $E(\mathbb{F}_q)$, constitutes the additive group $(\mathbb{G}, *)$. The base point P can be calculated as a function of the group G_{ID} as follows: consider that a group public identity, G_{ID}, then $Q_{ID} \in \mathbb{G}$ is to be generated by the Trusted Authority (TA) based on the group public identity: $Q_{ID} = \mathcal{H}(G_{ID})$, where $\mathcal{H}$ is a suitable hash function. Using its master key, s, the TA computes the base point P to be:

$$P = s * Q_{ID}$$

Every node i will then receive the point P from the TA and will generate its long term secret key s_i and compute its public key as:

$$P_i = s_i * P$$

The pair (s_i, P_i) is the long-term key of the participant i. All individual public keys should be available at the TA. The participating nodes will then get the public key of every node from the TA. These long term keys are used to implicitly authenticate the group members. These parameters should be chosen carefully and need to be validated to avoid many known attacks, as suggested for example in [45].

6.3 Group MTI Key Agreement Protocol (GMTI)

The protocol described in this section is an extension of one of the family of key agreement protocols originally proposed by Matsumoto *et al.* [47] in 1986. This family of protocols was designed to provide an implicit key authentication protocol, but not key confirmation. In this section, we explain how this family of protocols can be extended to a group of n participants (see Algorithm 6.1). GMTI considers that each member, M_j, can get an authentic copy of the public keys of the other group members ($P_i = s_i P$). The variable $r_i \in \mathbb{Z}_q$ is an ephemeral secret key and it is picked randomly for each run of the protocol.

Algorithm 6.1 *Authenticated GMTI Protocol*

Step 1 (Contributions collection): round i; $i \in [1, n-1]$

1. M_i *selects* $r_i \in \mathbb{Z}_p^*$
2. $M_i \xrightarrow{\{\frac{r_1 \ldots r_i s_1 \ldots s_i}{r_j s_j} P|_{j \in [1,i]}, r_1 \ldots r_i s_1 \ldots s_i P\}} M_{i+1}$

Step 2 (Key calculation): round n

3. M_n *selects* $r_n \in \mathbb{Z}_p^*$
4. $M_n \xrightarrow{\{\frac{r_1 \ldots r_n s_1 \ldots s_n}{r_i s_i} P|_{i \in [1,n-1]}\}} M_i$
5. *Upon receipt of the above,* M_i *computes:* $\frac{r_1 \ldots r_n s_1 \ldots s_n}{r_i s_i} r_i s_i P$
6. $K_n = \mathcal{F}(r_1 \ldots r_n s_1 \ldots s_n P)$

As an example, we illustrate the message flow during the protocol run for a group of four participants. In the first stage (contributions collections), each member generates a random value $r_i; i = 1, 2, 3$ and sends to its neighbour the following messages:

$$M_1 \rightarrow M_2 : r_1 s_1 P$$

$$M_2 \rightarrow M_3 : r_1 r_2 s_1 s_2 P, r_2 s_2 P, r_1 s_1 P$$

$$M_3 \rightarrow M_4 : r_1r_2r_3s_1s_2s_3P, r_1r_3s_1s_3P, r_2r_3s_2s_3P, r_1r_2s_1s_2P$$

In the last stage, M_4 randomly picks its ephemeral secret, r_4, and sends to the other members the following messages:

$$M_4 \rightarrow M_1 : r_2r_3r_4s_2s_3s_4P$$

$$M_4 \rightarrow M_2 : r_1r_3r_4s_1s_3s_4P$$

$$M_4 \rightarrow M_3 : r_1r_2r_4s_1s_2s_4P$$

By the end of this transmission, all the members are able to calculate the common secret $r_1r_2r_3r_4s_1s_2s_3s_4P$. The session key can be any function of the calculated common secret: $K_n = \mathcal{F}(r_1r_2r_3r_4s_1s_2s_3s_4P)$, where the function $\mathcal{F}$ can be a standard hash function for one or both point coordinates or a combination between both of the two coordinates.

6.3.1 Discussion

GMTI can be viewed as a direct extension of MTI/C(1), one of the MTI family for which there has been a heuristic proof of security so far. The authors had claimed that these protocols provide mutual implicit key authentication, but not key confirmation. In [45] Law, *et al.* describe a *small subgroup attack* for MTI/C(0) which can be applied also for MTI/C(1). This attack can be applied also for GMTI protocol, but it can be easily avoided by mandating the use of a base point P of prime order, and by applying validation processing by each member in the group to make sure that the received values lie in the subgroup of order q which is the order of the base point P.

Next, we will give an analysis of GMTI and a heuristic proof of its security. All the messages transmitted through the protocol can be represented as $\frac{\prod r_is_i}{r_is_i}P; i \in [1, n]$, where n is the number of members in the group. Assuming the intractability

of the elliptic curve discreet logarithm problem, an intruder will not be able to calculate the generated shared secret, even if the intruder has access to any long-term key s_i of an authorized group member, M_i. The possible attack if the intruder has access to a long-term key of any member is as follows: suppose that s_i has been compromised, the intruder can strip this value from the message transmitted by M_i and add another value s_I. As a result of this attack, all the group members except M_i will calculate a secret value $r_1r_2...r_ns_1s_2...s_I...s_nP$, while M_i will calculate $r_1r_2...r_ns_1s_2...s_nP$. Meanwhile the intruder will not be able to calculate any of the keys since the intruder has no access to the ephemeral secret key r_i of M_i. This attack can be easily discovered, by the fulfillment of the key confirmation property. Similar conclusion can be reached using the formal analysis model of chapter 5, which can be stated as follows: the long-term key is never transmitted alone or in plain form in such a way that knowing a long-term key of one or more members does note enable the intruder to get access to the generated session key.

From the complexity point of view, GMTI protocol has the following characteristics

Rounds	$R = n$
Messages	$M = 2(n-1)$
Exponentiations/M_i	$Ex_i = i + 1$
Exponentiations/M_n	$Ex_n = (n-1)$
Maximum Message size	$M_s = n - 1$

From the above table, we can see that the complexity of the GMTI protocol is a linear function of the group size. Also we can see that the last two members in the group are performing the largest number of computations.

6.4 Group MQV Protocol (GMQV)

In this section, we present another authenticated key agreement protocol (see Algorithm 6.2). The main protocol consists of two stages: in the first stage, the contribution of all group members are collected, while in the second stage, the last member in the group sends a unicast message to each group member which enables them to calculate a common session key. GMQV follows the same steps as that of GMTI in collecting the members' contributions, but the contents of the messages are different and the final generated session key is different than that of GMTI. The most interesting property of this protocol and GMTI is that the authentication property is added within the steps of key agreement protocol, i.e. there is no extra manipulated messages. The only difference between GMQV and GMTI is that the contributions are collected in a separate format, i.e. the ephemeral contributions are collected separate from the long-term shares. As a result, the message size of GMQV is greater than that of GMTI.

6.4.1 Protocol Description

In the following description , S_A and s_a are the member $A's$ long-term public and private keys respectively, while R_A and r_a are the public and private ephemeral keys. For illustration, we will describe the steps of the protocol for a group of four members, but it can be applied for any group size. The group members will be M_1, M_2, M_3, M_4, where M_4 is the group leader. The first step consists of 3 rounds, in each round each member calculates and sends its public share using its long-term and ephemeral secret key. For example

$$M_1 \rightarrow M_2 : r_1 P$$

$$M_2 \rightarrow M_3 : r_1 r_2 P, s_1 s_2 P, r_2 P, r_1 P$$

$$M_3 \rightarrow M_4 : r_1r_2r_3P, r_1r_3P, r_2r_3P,$$

$$r_1r_2P, s_1s_2s_3P, s_1s_3P, s_2s_3P, s_1s_2P$$

Note that M_1 does not need to send its long-term public share since it should be publicly available. In the last round M_4 sends the following messages to the other group members.

$$M_4 \rightarrow M_1 : r_2r_3r_4P, r_4s_2s_3s_4P$$

$$M_4 \rightarrow M_2 : r_1r_3r_4P, r_4s_1s_3s_4P$$

$$M_4 \rightarrow M_3 : r_1r_2r_4P, r_4s_1s_2s_4P$$

By the end of this round, all the group members are able to calculate the same two points ($r_1r_2r_3r_4P$, $r_4s_1s_2s_3s_4P$). The generated common secret can be any function of these two points:

$$K_n = \mathcal{F}(r_1r_2r_3r_4P, r_4s_1s_2s_3s_4P)$$

Algorithm 6.2 *Group MQV Protocol*

Step 1 (contributions collection):

round i; $I \in [1, n-1]$

1. M_i *selects* $r_I \in \mathbb{Z}_p^*$
2. $M_i \xrightarrow{\{\frac{r_1 \ldots r_i}{r_j}P, \frac{s_1 \ldots s_i}{s_j}P|_{j\in[1,i]},\ r_1 \ldots r_iP,\ s_1 \ldots s_iP\}} M_{i+1}$

Step 2 (Key calculation): round n

3. M_n *selects* $r_n \in \mathbb{Z}_p^*$
4. $M_n \xrightarrow{\{\frac{r_1 \ldots r_n}{r_i}P, \frac{r_n s_1 \ldots s_n}{s_i}P|_{i\in[1,n-1]}\}} M_i$
5. *Upon receipt of the above,* M_i *computes:* $\frac{r_1 \ldots r_n}{r_i}r_iP, \frac{r_n s_1 \ldots s_n}{s_i}s_iP$
6. $K_n = \mathcal{F}(r_1 \ldots r_nP, r_n s_1 \ldots s_nP)$

The suggested possibilities for the function $\mathcal{F}$ can be exclusive OR, Weil or Tate pairings, hashing or any function of the coordinates of the two generated points.

6.4.2 Discussion

GMQV protocol can be considered as an extension to one of the recent authenticated two-party key agreement protocols (MQV) presented in [45], which is another way of combining the long-term key and the ephemeral key to provide the implicit key authentication. In the following, we will give a heuristical proof of the security of the GMQV protocol. We can see from the analysis of the protocol that all the messages transmitted through the protocol can be represented as $\frac{\prod s_i}{s_i}P; i \in [1,n]$ and $\frac{\prod r_i}{r_i}P; i \in [1,n]$, where n is the number of members in the group. Assuming the intractability of the elliptic curve discrete logarithm problem, an intruder will not be able to calculate the generated shared secret, even if the intruder has access to a long-term key s_i of an authorized group member M_i. The only possible attack can be done as follows: suppose that s_i has been compromised, the intruder can strip this value from the message transmitted by M_i and add another value s_I (I here relates to Intruder). The result from this attack will be as following: all the group members except M_i will calculate a secret value $\mathcal{F}(r_1r_2...r_n, r_ns_1s_2...s_I...s_nP)$, while M_i will calculate $\mathcal{F}(r_1r_2...r_n, r_ns_1s_2...s_nP)$. Meanwhile, the intruder will not be able to calculate either key since the intruder has no access to the ephemeral secret key r_i of M_i. This attack can be easily discovered, by fulfillment key confirmation property. The formal analysis for GMQV is a straightforward (as that of GMTI) and can be easily applied. The result should be the same as the heuristic analysis.

The characteristics of GMQV are as follows:

Rounds	$R = n$
Messages	$M = 2(n-1)$
Exponentiations/M_i	$Ex_i = 2(i+2)$
Exponentiations/M_n	$Ex_n = (n-1)$
Maximum Message Size	$M_s = 2(i+1)$

6.5 Remarks

The security of GMTI and GMQV protocols is comparable, since both of them follow the same idea of combining the long-term key with the ephemeral key to provide authentication. However, each protocol uses a different way to fulfill this combination, which let GMQV more flexible in calculating the session key in the last step of the protocol. This flexibility gives more randomness and a wide choice in calculating the session key. From the complexity point of view, we can see that the number of rounds and the number of required messages are equal in both protocols. However, GMTI is much better in both computation and maximum message size. The maximum message size of GMQV protocol is greater than that of GMTI and also the required exponentiation per member in GMQV is greater than that of GMTI. Also the generated secret from GMQV required another computation process, like the Weil pairing, which is costly.

The previous discussion of both GMTI and GMQV protocols shows that the overhead of the key agreement protocols (both communication and computation) increases linearly with the size of the communicating group , which is not efficient. Also GMTI and GMQV protocols suppose that the group members are distributed in a ring, which is not suitable for applications such as ad hoc networks. Both protocols can be used as building blocks in the framework discussed in Chapter 3. The resulting protocols will be efficient and secure against active attacks.

6.6 Authenticated BD Protocol (ABD)

In some applications such as mobile networks, the distribution of members in a fixed logical structure is not preferable. In this section, a different setting based on a multiplicative group will be presented. The proposed protocol, ABD, (Algorithm 6.3) does not require a fixed topology for the underlying structure. However the group members should have the capability to broadcast messages to the group members, and receive a simultaneous broadcasts as well. These required characteristics may not be practically available in many environments. The presented protocol is an authenticated version of the BD protocol [19], which runs only in two rounds regardless of the number of participants in the communicating group. In this section, we propose a new authenticated key agreement protocol based on the idea used in the BD protocol. The proposed protocol ABD provides authentication with the same primitives as the key agreement protocol as discussed before with a minimum increase in communication and computation overhead.

The protocol assumes that each member of the group has a long-term key pair (s_i, α^{s_i}) where $i = 1, \ldots, n$. The protocol runs as follows:

1. Each member, M_i, picks a random number, r_i, calculates its ephemeral public key, α^{r_i}, and broadcasts this value to the other members.

2. Upon receiving the ephemeral public key of the other members, each member, M_i, calculates $X_i = \alpha^{r_i s_{i-1}}$ and $Y_i = \alpha^{r_{i+1} s_i}$. From these two values, each member calculates and broadcasts $Z_i = \frac{Y_i}{X_i}$ to the other group members.

3. Upon receiving these messages, each M_i can check the correctness of the received messages by making sure that $X_i = Y_i \Pi Z_j|_{j \in [1,n], j \neq i}$. If the check fails, M_i terminates the protocol runs and sends an error message to the other group members. Otherwise, M_i calculates his view of the common secret as

follows:

$$K_i = Y_i^n Z_{i+1}^{n-1} Z_{i+2}^{n-2} \ldots Z_{i-1}$$

To demonstrate the protocol, an example of a group consisting of four members is considered. In the first step, each member, M_i, calculates X_i, Y_i and Z_i as shown below:

1. $X_1 = \alpha^{r_1 s_4}, Y_1 = \alpha^{r_2 s_1}, Z_1 = \frac{Y_1}{X_1} = \frac{\alpha^{r_2 s_1}}{\alpha^{r_1 s_4}}$
2. $X_2 = \alpha^{r_2 s_1}, Y_2 = \alpha^{r_3 s_2}, Z_2 = \frac{Y_2}{X_2} = \frac{\alpha^{r_3 s_2}}{\alpha^{r_2 s_1}}$
3. $X_3 = \alpha^{r_3 s_2}, Y_3 = \alpha^{r_4 s_3}, Z_3 = \frac{Y_1}{X_1} = \frac{\alpha^{r_4 s_3}}{\alpha^{r_3 s_2}}$
4. $X_4 = \alpha^{r_4 s_3}, Y_1 = \alpha^{r_1 s_4}, Z_1 = \frac{Y_1}{X_1} = \frac{\alpha^{r_1 s_4}}{\alpha^{r_4 s_3}}$

Upon receiving $Z_j; j \neq i$, each member M_i can calculate its view of the key $S_n(M_i)$

$$S_n(M_1) = Y_1^4 Z_2^3 Z_3^2 Z_4 = \alpha^{4r_2 s_1} \frac{\alpha^{3r_3 s_2}}{\alpha^{3r_2 s_1}} \frac{\alpha^{2r_4 s_3}}{\alpha^{2r_3 s_2}} \frac{\alpha^{r_1 s_4}}{\alpha^{r_4 s_3}}$$

$$S_n(M_2) = Y_2^4 Z_3^3 Z_4^2 Z_1 = \alpha^{4r_3 s_2} \frac{\alpha^{3r_4 s_3}}{\alpha^{3r_3 s_2}} \frac{\alpha^{2r_1 s_4}}{\alpha^{2r_4 3 s_3}} \frac{\alpha^{r_2 s_1}}{\alpha^{r_1 s_4}}$$

$$S_n(M_3) = Y_3^4 Z_4^3 Z_1^2 Z_2 = \alpha^{4r_4 s_3} \frac{\alpha^{3r_1 s_4}}{\alpha^{3r_4 s_3}} \frac{\alpha^{2r_2 s_1}}{\alpha^{2r_1 s_4}} \frac{\alpha^{r_3 s_2}}{\alpha^{r_2 s_1}}$$

$$S_n(M_4) = Y_4^4 Z_1^3 Z_2^2 Z_3 = \alpha^{4r_1 s_4} \frac{\alpha^{3r_2 s_1}}{\alpha^{3r_1 s_4}} \frac{\alpha^{2r_3 s_2}}{\alpha^{2r_2 s_1}} \frac{\alpha^{r_4 s_3}}{\alpha^{r_3 s_2}}$$

We can easily check that:

$S_n(M_1) = S_n(M_2) = S_n(M_3) = S_n(M_4) = \alpha^{r_1 s_4 + r_2 s_1 + r_3 s_2 + r_4 s_3}$.

To provide an additional measure of security, the protocol can have the members use any suitable function $f(S_n)$ of the generated session key S_n, such as those offered by the hash function family.

Algorithm 6.3 *Authenticated BD Protocol (ABD)*

Round 1*: Contributions Collection:*

1. M_i *selects* $r_i \in \mathbb{Z}_p^*$
2. $M_i \xrightarrow{\{\alpha^{r_i}|_{i,j\in[1,n],i\neq j}\}} M_j$
3. M_i *calculates* $X_i = \alpha^{r_i s_{i-1}}, Y_i = \alpha^{r_{i+1} s_i}$, *and* $Z_i = \frac{Y_i}{X_i}$
4. $M_i \xrightarrow{Z_i|_{i,j\in[1,n],i\neq j}} M_j$

Round 2*: Key Calculation:*

5. *If* $X_i = Y_i \Pi Z_j|_{j\in[1,n],j\neq i}$
6. *Then* $K_i = Y_i^n Z_{i+1}^{n-1} Z_{i+2}^{n-2} \ldots Z_{i-1}$
7. *Else Send Failure message*

6.6.1 Discussion

In this section, a discussion of the proposed protocol (ABD) in terms of its complexity and also a heuristic security analysis for the protocol is presented. From the complexity point of view, it is clear that the ABD protocol is more efficient than both GMTI and GMQV in terms of number of rounds and number of required messages. The protocols runs in just two rounds regardless of the group size. Regarding the total number of messages, each member in the group makes two broadcasting messages. Regarding the computation cost, every member in the group performs 3 modular exponentiations and one inversion regardless of the number of members in the group. The message size in the protocol is fixed and can be normalized as 1 since it is irrelevant to the group size. It should be mentioned that ABD is more suitable for peer groups since the rule and the responsibility of all members are the same. In other words, there is no special rule of any member and there is no need for arranging the group members into a fixed topology, since each member broad-

cast its message to the other group members wherever their place in the network. However, each member should knows its one hop away neighbours. Additionally, we have to mention that the ABD protocol based on the ability of each member to broadcast a message to the rest of the group members and to receive a simultaneous messages from all other group members. These capabilities may not be a available in some applications. The characteristics of the protocol are summarized in the following table

Rounds	$R = 2$
Messages	$M = 2n$
Exponentiations/M_i	$Ex_i = 3$
Inversions/M_i	$INV_i = 1$
Maximum Message Size	$M_s = 1$

In the following, we consider a heuristic proof of our proposed protocol against passive and active attacks. If we consider the kind of messages transmitted during the protocol conduction, we have two kind of messages: α^{r_i} and $Z_i = \frac{\alpha^{r_{i+1}s_i}}{\alpha^{r_i s_{i-1}}}$. It is clear that it is difficult for any intruder to extract any information from these two types of messages. Even if the intruder has access to a long-term key of any member, he will not be able to get any benefit from knowing this long-term key, since the long term key of any member is transmitted in combination with the ephemeral key of the member in such a way that it is so difficult to decouple the two keys. The only way is to solve the difficult DLP. As a result, we can claim that the ABD protocol provides implicit key authentication, since no one outside the authorized members can get access to the shared generated secret or any partial information about it. Also, we can claim that the ABD provides Perfect Forward Security (PFS), since knowing a long-term key of any member dos not divulge any partial information about the generated secret. It is clear that the generated secret is independent of a

previously generated secret in a previous session, so the key independence (forward and backward) is satisfied in the ABD protocol.

6.7 Summary

Design a secure group key agreement should be accompanied with a proper authentication scheme. As it was shown in the previous chapter, using an improper authentication scheme gives chances to an active intruder to attack the main group key agreement protocols. In this chapter, different solutions for authenticated group key agreement protocols have been presented. The first two protocols (GMTI and GMQV) are suitable where the communications overhead is not an important issue, as is the case with like in LANs. Also they can be applied in static networks with a fixed topology. While the third protocol, ABD, is efficient in terms of the number of rounds and communications cost. However, the ABD protocol requires that all the group members are equipped with broadcasting capabilities. A heuristic proof of security of each protocol is also given. These proofs supported the finding of chapter 5, using the proposed formal model. An important feature in the protocols presented is that the authentication scheme is provided using the same primitives used in conducting the group key agreement protocols. As a result, these protocols provide authenticated key agreement protocols with comparable complexity to the previous unauthenticated protocols.

Chapter 7

Conclusion and Future Work

This chapter gives a summary of the work done in this thesis. It also gives an outlook on open problems and some possible research directions for future work.

Group key agreement protocols are a crucial issue in group communication security. While the two-party key agreement has been extensively investigated and well-understood, less attention has been dedicated to n-party key agreement protocols. The extension of two-party protocols into group key agreement protocols needs to be done with caution due to the many different characteristics. Moreover, efficiency becomes an important issue, since the generated protocols are multi-round protocols.

In this thesis, we have investigated the problem of key agreement in group communications, where we have provided an extended survey of related work and important theoretical and practical related issues. Our discussion clearly identifies two main (related and conflicting) problems in group key agreement protocols: efficiency and security. To tackle the first problem, efficiency, we have proposed adoption of clustering techniques, which are natural in many applications, to distribute the members of the group in a logical hierarchical scheme. The use of differ-

ent clustering schemes which conform to the requirements of group communication has been suggested.

Using this approach, two efficient protocols have been proposed, HGKM and CGKA. In each protocol, we have explained how the adoption of clustering techniques improves the performance of the proposed protocols compared to the existing protocols. Additionally, we have explained that this adoption allows for more flexibility with networks that consists of heterogeneous systems. Also, discussed is the fact that confining the size of clusters to a certain small size increases the reliability of the protocol, since most of the transmitted messages will be within a certain geographical distance. This can be proved to be beneficial in applications such as in ad hoc wireless networks, where one-hop messages are more reliable than multi-hop messages. Our protocols also provide a greater degree of fault tolerance, as a link failure or compromise of any member in a flat setting will affect all the group members, whereas in a hierarchical structure a link or member failure affects only the cluster which this link or member belongs to. Advantages were also gained in auxiliary protocols, as only a subset of the group members are involved in the regeneration of the group session key.

HGKM and CGKA protocols are not suitable for use in ad hoc wireless networks due to their special characteristics. In chapter 4, we proposed a modified group key agreement protocol, DTGDH, where the distribution of clusters through the network is arbitrary and flexible to accommodate the dynamic nature of these networks. We have showed that our proposed protocol provides an easy solution to the members mobility. We have also given an authenticated version of our proposed protocol, A-DTGDH, using identity based pairwise shared key for authentication. Using identity based pairwise key provides mutual authentication without exchanging any messages and without relying on any central authorities.

Regarding the protocol security, there has been a common belief that the secu-

rity of group key agreement protocols can be guaranteed directly as an extension of the security of two-party key agreement. This assumption cannot be guaranteed in all cases, due to the different nature of group communications. We have proposed a simple model to analyze the security of Group Diffie-Hellman protocols. In the analysis of one of this family of protocols, the Cliques suite, we have found several practical attacks against different security requirements.

Our extended survey have shown that existing proposals use different primitives for authentication such as digital signatures or message authentication code (MAC) than those used for key agreement protocols. This can potentially lead to large overhead, making it impractical for many applications.

We have proposed different solutions to provide authentication to group key agreement protocols using the same cryptographic primitives. The first two protocols, GMTI and GMQV, use the set of points on an elliptic curve as its underlying algebraic group and are suitable for static small size networks with a fixed topology. The third one, ABD, uses a multiplicative group as their underlying algebraic group and it is suitable for larger dynamic networks with any topology such as ad hoc wireless networks. We have explained the differences among these protocols and the suitable environment where each protocol can efficiently fit.

The overall contribution of this thesis has been to provide a framework of group key agreement protocols which is efficient, scalable, flexible and provably secure, which has been achieved.

Observations through this thesis suggest a number of lines for future work. First, although the efficiency of the proposed framework has been significantly improved, more improvement is required to accommodate larger networks. Second, implementation of the proposed framework is required to ascertain which cluster size and number of levels are more suitable for a given number of participants in certain environment and provides indication of its performance. Also, this implementation

will give an indication about the effect of these protocols on the Quality of services (QoS) of the underlying group communication systems, especially in environment wherein QoS is an critical issue such as mobile networks.

The security of the proposed framework based on DHP or generally DLP, other primitives may give a better performance and/or more security. Also, incorporation of the new certificateless approach, which is advantageous in security of ad hoc networks, within our framework may improve the performance of our framework.

From our discussion of formal analysis techniques, we have found some models proved the security of certain protocols, while others disproved the security of the same protocols. This conflict affects the trustworthiness of these methods. Unified formal analysis models are crucial nowadays to give some confidence in the security of cryptographic protocols to vendors.

In this thesis, we have proposed several clustering techniques, which lead to improvement in overall performance. Other clustering techniques may also provide solutions to the scalability problem, more flexibility and/or provides a better handling for the network failures.

Bibliography

[1] FIPS Publication 186, *"Digital Signaure Standards"*, National Institute of Standard and Technology, May 1994, Available at http://csrc.nist.gov/fips.

[2] M. Abadi and P. Rogaway, *"Reconciling Two Views of Cryptography"*, Journal of Cryptology **15** (2002), no. 2, 103–127.

[3] A. Abdel-Hafez, A. Miri, and L. Orozco-Barbosa, *"Scalable and Fault-Tolerant Key Agreement Protocols for Dynamic Groups"*, Submitted to Journal of Computer Communications.

[4] N. Asokan and P. Ginzboorg, *"Key Agreement in Ad-hoc Networks"*, Journal of Computer Communications **23** (2000), no. 17, 1627–1637.

[5] G. Atenies, M. Steiner, and G. Tsudik, *"New Multiparty Authentication Services and Key Agreement Protocols"*, IEEE Journal on Selected Areas in Communications (JSAC) **18** (2000), no. 4, 628–639.

[6] S. Banerjee and B. Bhattcharjee, *"Scalable Secure Group Communication over IP Multicast"*, IEEE Journal of Selected Areas in Communications (JSAC), Special Issue On Network Support for Group Communication **20** (2002), no. 8, 1511–1527.

[7] S. Banerjee and S. Khuller, *"A Clustering Scheme for Hierarchical Control in Multi-Hop Wireless Networks"*, In the Proc. of IEEE INFOCOM'01, pp. 1028–1037 (2001).

[8] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti, *"Secure Pebblenets"*, the Proc. of 2001 ACM International Symp. on Mobile Ad Hoc Networking and computing, pp. 156–163 (Long Beach, CA, USA), Oct. 2001.

[9] K. Becker and U. Wille, *"Communication Complexit of Group Key Distribution "*, the Proc. of ACM CCS'98, pp. 1–6, 1998.

[10] M. Bellar and P. Rogaway, *"Entity Authentication and Key Distribution"*, the Proc. of CRYPTO'93 the 13th Annual International Conference, pp. 232–249, LNCS 773, 1994.

[11] M. Bellare, D. Pointcheval, and P. Rogaway, *"Authenticated Key Exchange Secure Against Dictionary Attacks"*, In Advances in Cryptology–EUROCRYPT'00 **LNCS 1807** (2000), 139–155, Full version of this paper available from http://www-cse.ucsd.edu/users/mihir.

[12] M. Bellare and P. Rogaway, *"Random Oracles are Practical: a Paradigm for Designing Efficient Protocols"*, In the Proc. of the 1st ACM Conference on Computer and Communication Security, pp. 62–73 (1993).

[13] S. Blake-Wilson, C. Johnson, and A. Menezes, *"Key Agreement Protocols and Their Security Analysis"*, the Proc. of the 6th IMA International Conference on Cryptography and Coding, pp. 30–45, LNCS 1355, 1997.

[14] S. Blake-Wilson and A. Menezes, *"Authenticated Diffie-Hellman Key Agreement Protocols"*, In The Proc. of the 5th Annual Workshop on Selected Areas in Cryptography (SAC'98), pp. 339–361 **LNCS 1556** (1998).

[15] D. Boneh, *"The Decision Diffie-Hellman Problem"*, The Proc. of the 3rd Algorithmic Number Theory Symposium (ANTS'98) (Berlin Germany), LNCS 1423, Springer-Verlag, 1998, Invited Paper.

[16] E. Bresson, O. Chevassut, and D. Poincheval, *"Provably Authenticated Group Diffie-Hellman Key Exchange–The Dynamic Case"*, In the Proc. of 18th Conference on Computer and Communication Security, Advances in Cryptology SIACRYPT'01, pp. 290–309 LNCS **2248** (2001).

[17] E. Bresson, O. Chevassut, D. Poincheval, and J. Quisquater, *"Provably Authenticated Group Diffie-Hellman Key Exchange"*, In The Proc. of the 8th Conference om Computer and Communication Security (2001), 255–264.

[18] B. Briscoe, *"MARKS: Zero Side-Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences"*, the Proc. of the 1st International Workshop on Networked Group Communication, pp. 301–320, Nov. 1999.

[19] M. Burmester and Y. Desmedt, *"A Secure and Efficient Conference Key Distribution Systems"*, tThe Proc. of Advances in Cryptology EUROCRYPT'94, pp. 275–286, LNCS 950, 1994.

[20] M. Burrows, M. Abadi, and R. Needham, *"A Logic of Authentication"*, ACM Transactions on Computer Systems **8** (1990), no. 1, 18–36.

[21] R. Canetti, G. Tsudik, G. Itkis, D. Micciano, M. Naor, and B. Pinkas, *"Multicast Security: A Taxonomy and Efficient Construction"*, IEEE/ACM Transactions on Networking **8** (2000), no. 1, 16–30.

[22] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, *"Key Management for Secure Internet Multicast Using Boolean Minimization Techniques"*, the Proc. of IEEE Infocom'99, pp. 689–698, vol. 2, Mar. 1999.

[23] M. Chatterjee, S.K. Das, and D. Turgut, *"WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks"*, Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks) **5** (2002), no. 2, 193–204.

[24] I. Cidon and O. Mokryn, *"Propoagation and Leader Election in Multihop Broadcast Environment"*, the Proc. of the 12th International Symposium on DIStributed Computing (DISC98), pp. 104–119 (Greece), Sep. 1998.

[25] R. Cramer and V. Shoup, *"A Practical Public Key Cryptosystem Provable Secure Against Adaptive Chosen Ciphertext Attacks"*, In the Proc. of Advances in Cryptology Crypto'98, pp. 13–25 (1998), no. LNCS 1460.

[26] B. Dahill, B. Levine, E. Royer, and C. Shields, *"A Secure Routing Protocol for Ad Hoc Networks"*, Technical Report UM-CS-2001-037, University of Massachusetts, Aug. 2001.

[27] D. Denning and D. K. Branstad, *"A Taxonomy for Key Escrow Encryption Systems"*, The Communications of the ACM **39** (1996), no. 3, 34–40.

[28] W. Diffie and M. Hellman, *"New Direction in Cryptography"*, IEEE Transactions on Information Theory **28** (1976), no. 5, 644–654.

[29] M. Franklin and D. Boneh, *"Identity Based Encryption from the Weil Pairing"*, the Proc. of Advances in Cryptology – CRYPTO'01, pp. 514–532, LNCS 2248, 2001.

[30] S. Goldwasser and S. Micali, *"Probabilistic Encryption"*, Journal of Computer and System Sciences **28** (1984), 270–299.

[31] C. G. Gunther and A. B. Boveri, *"An Identity-Based Key-Exchange Protocol"*, th proc. of Advances in Cryptology– CRYPTO'01, pp. 29–37, LNCS 434, 1990.

[32] Jiawei Han and Michelle Kamber, *"Data Mining: Concepts and Techniques"*, Morgan Kaufmann series, Morgan Kaufmann Publishers, 2001.

[33] H. Harney and C. Muckenhirn, *"Group Key Management Protocol (GKMP) Specification"*, Network Working Group RFC 2093 (Experimental), July 1997, Internet Engineering Task Force.

[34] Y. Hu, A. Perrig, and D. Johnson, *"Ariadne: A secure on-demand routing protocol for ad hoc networks"*, the Proc. of the 8th ACM International Conference on Mobile Computing and Networking MobiCom'02, 2002.

[35] ———, *"SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks"*, the Proc. of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02), pp. 3–13, Jun. 2002.

[36] Mohammed Ilyas, *"The Handbook of Ad Hoc Wireless Networks"*, CRC Press, Washington D.C., 2003.

[37] I. Ingemarsson, D. Tang, and C. Wong, *"A Conference Key Distribution System"*, IEEE Transactions on Information Theory **28** (1982), no. 5, 714–720.

[38] M. Just and S. Vauenay, *"Authenticated Multi-Party Key Agreement"*, In Proc. of Advances in Cryptology –EURCRYPT'96, International Conference on the Theory and Application of Cryptology (1996), no. LNCS 1070.

[39] B. S. Kaliski, *"An Unknown Key-Share Attack on the MQV Key Agreement Protocol"*, ACM TRansactions on Information and System Security **4** (2001), no. 3, 275–288.

[40] J. Katz and Moti Yung, *"Scalable Protocols for Authenticated Group Key Exchange"*, the Pro. of the 23rd Annual International Cryptology Conference CRYPTO 2003, pp. 630-648, LNCS 2656, Aug. 2003.

[41] A. Khalili, J. Katz, and W. A. Arbaugh, *"Towards Secuity solutions for Trully Ad Hoc Networks"*, the Proc. of IEEE Workshop on Security and Assurance in Ad Hoc Networks, in Conjuction with the 2003 International Symposium on Applications and the Internet, Jan. 2003.

[42] Y. Kim, D. Mazzochi, and G. Tsudik, *"Admission Control in Peer Groups"*, In the Proc. of IEEE International Symposium on Network Computing and Applications, pp. 131–140 (2003).

[43] Y. Kim, A. Perrig, and G. Tsudik, *"Simple and Fault-Tolerant Key Agreement for Dynamic Collaborarive Groups"*, In the Proc. of 7th ACM Conference on Computer Communication Security (ACM CCS' 2000), pp. 235–244 (2000).

[44] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, *"An Efficientr Protocol for Authenticated Key Agreement Protocol"*, Technical Report CORR 98–5, University of Waterloo, Waterloo, Canada, Mar. 1998.

[45] ———, *"An Efficientr Protocol for Authenticated Key Agreement Protocol"*, Design, Codes and Cryptography **28** (2003), no. 2, 119–134.

[46] S. Lee, S. Hong, H. Yoon, and Y. Cho, *"Accelerating Key Establishment Protocols for Mobile Communication"*, the Proc. of 4th Australasian Conference, ACISP'99, pp. 51–63, LNCS 1587, 1999.

[47] T. Matsumoto, Y. Takashima, and H. Imai, *"On Seeking Smart Public-Key Distribution Systems"*, The Transactions of the IECE of Japan **E. 69** (1986), no. 2, 99–106.

[48] C. Meadows, *"The NRL Protocol Analyzer : An Overview"*, Journal of Logic Programming **26** (1996), no. 2, 113–131.

[49] ———, *"Extending Formal Cryptographic Protocol Analysis Techniques for Group Protocols and Low-Level Cryptographic Primitives "*, In the Proc. of the Workshop on Issues in the Theory of Security (WITS'00) (2000).

[50] ———, *"What Makes a Cryptographic Protocols Secure? The Evolution of Requirements Specification in Formal Cryptographic Protocol Analysis"*, the Proc. of ESOP, pp. 10–21, LNCS 2618, Apr. 2003.

[51] A. Menezes, M. Qu, and S. Vanstone, *"Some New Key Agreement Protocols Providing Implicit Authentication"*, the Proc. of the 2nd Workshop on Selected Areas in Cryptography (SAC '95), pp. 22–32 (Ottawa), May 1995.

[52] A. J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *"Handbook of Applied Cryptography"*, CRC Press, 1997.

[53] S. Mittra, *"Iolus : A Framework for Scalable Secure Multicasting"*, the Proc. of the ACM SIGCOMM'97 Conference, pp. 277–288, Oct. 1997.

[54] S. Mullender, *"Distributed Systems"*, 2nd. ed., ACM Press frontier series, Addison-Wesley Pub. Co., 1993.

[55] K. Nyberg and R. Rueppel, *"A New Signature Scheme Based on DSA Giving Message Recovery"*, the Proc. of the 1st ACM Conference on Computer and Communications Security, pp. 58–61, Nov. 1993.

[56] K. Ohgishi, R. Sakai, and M. Kasahara, *"Cryptosystems Based on Pairing"*, the Proc. of the Symposium on Cryptgraphy and Infrmation Security (SCIS2000) (Okinawa, Japan), Jan. 2000.

[57] M. Pease, R. Shostak, and L. Lamport, *"Reaching Agreement in the Presence of faults"*, Journal of ACM **27** (1980), no. 2, 228–234.

[58] O. Pereira, *"Modelling and Security Analysis of Authenticated Group Key Agreement Protocols"*, Ph.d. thesis, Univerite Catholique De Louvain, Faculte Des Sciences Appliquees, Louvain-la-Neuve, Belgique, 2003.

[59] O. Pereira and J. J. Quisquater, *"A Security Analysis of the Cliques Protocol Suites"*, the Proc. of the 14-th IEEE Computer Security Foundations Workshop, pp. 73–81, June 2001.

[60] ———, *"Some Attacks upon Authenticated Group Key Agreement Protocols"*, To appear in Journal of Computer Security (2003).

[61] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, *"SPINS: Security Protocols for Sensor Networks"*, the Proc. of the 7th Annual ACM International Conference on Mobile Computing and Networks (Mobicom'01), pp. 189–199, July 2001.

[62] R. Rueppel and P. van Oorschot, *"Modern Key Agreement Techniques"*, Computer Communication Journal **17** (1994), no. 7, 458–465.

[63] A. Shamir, *"Identity Based Cryptosystems and Signature Schemes"*, the Proc. of Advances in Cryptology – Crypto'84, LNCS 196, 1984.

[64] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener, *"A Secure Audio Teleconference System"*, the Proc. of the Advances in Cryptology–CRYPTO'88, pp. 520–528, LNCS 403, Aug. 1990.

[65] M. Steiner, G. Tsudik, and M. Waidner, *"Diffie-Hellman Key Distribution Extended to Group Communication"*, In the Proc. of ACM CCS'96 (1996).

[66] ———, *"Cliques: A New Approach to Group Key Agreement"*, The Proc. of the 18th International Conference on Distributed Computing Systems (ICDCS'98), pp. 380–387, May 1998.

[67] ———, *"Key Agreement in Dynamic Peer Groups"*, IEEE Transactions on Parallel and Distributed Systems **11** (2000), no. 8, 769–780.

[68] P. F. Syverson and P. C. van Oorscchot, *"On Unifying Some Cryptographic Protocol Logics"*, In Proc. of the IEEE Symposium on Research in Security and Privacy, pp. 14–24 (1994).

[69] S. Tsuji and T. Itoh, *"An ID-Based Cryptosystems Based on The Discrete Logarithm Problem"*, IEEE Journal on Selected Areas in Communication 7 (1989), no. 4, 467–473.

[70] P. C. van Oorrschot, *"Extending Cryptographic Logics of Belief to Key Agreement Protocols"*, In the Proc. of the 1st ACM Conference on Computer and Communication security, pp. 232–243 (1993).

[71] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, *"The VersaKey Framework: Versatile Group Key Management"*, IEEE Journal on Selected Areas in Communications **17** (1999), no. 9, 1–15.

[72] D. M. Wallner, E. J. Harder, and R. C. Agee, *"Key Managment for Multicast : Issues and Archetictures"*, Internet Draft Informational RFC–2627, June, 1999.

[73] L. C. Washington, *"Elliptic Curves: Number Theory and Cryptography"*, Discrete Mathematics and Its Applications, Chapman Hall/CRC, 2003.

[74] C. K. Wong, M. Gouda, and S. S. Lam, *"Secure Group Communications Using Key Graphs"*, IEEE/ACM Transactions on Networking **8** (2000), no. 1, 16–30.

[75] Y. Yacobi and Z. Shmuely, *"On Key Distribution Systems"*, the Proc. of Advances in Cryptology - CRYPTO'89, pp. 344–355 (Santa Barbara, CA, USA) (Giles Brassard, ed.), LNCS 435, 1989.

[76] L. Zhou and Z. Haas, *"Securing Ad Hoc Networks"*, IEEE Network Magazine **13** (1999), no. 6, 24–30.

Appendix A

Diffie-Hellman Problem

The security of most public-key cryptosystems relies on the apparent intractability of certain computational problems. The true computational complexities of these problems are not known. That is to say, they are believed to be intractable, although their intractability is not yet proven. The most common computational problems used in cryptology are Integer Factorization Problem (IFP), Quadratic Residuosity Problem (QRP), Subset Sum Problem (SSP), and Discrete Logarithm Problem (DLP). Each problem has its related cryptographic problem. For example Diffie-Hellman Problem (DHP) is related to DLP problem. The common feature of these problems is that they are candidates to One-Way function (OWF). The basic notion is that values of the function are easy to calculate given an input in the domain, but that given a value in the range it is difficult to find an input that would map to that value (even if many exist).

A.1 The Discrete Logarithm Problem

When working with the real numbers, exponentiation (finding b^x) is not significantly easier than the inverse operation (finding $\log_b x$). For a finite group ($\mathbb{Z}_{\mathbb{P}}^*$ for example), computation of b^x for a large x is fairly faster ($O(\log x)$). In the other hand, given $y = b^x$, finding $x = \log_b y$ is not an easy task. This process is called "Discrete Logarithm Problem." The word "discrete" distinguishes the finite group from the classical (continues) situation. Formally, the discrete logarithm problem (DLP) can be defined as following:

Definition A.1 *Given a prime p, a generator α of $\mathbb{Z}_p^*$ and an element $\beta \in \mathbb{Z}_p^*$, find the integer $x, 0 \leq x \leq p-2$, such that $\alpha^x \equiv \beta \pmod p$*

There have been many attempts to solve the Discrete Logarithm Problem. Some of these algorithms works in our arbitrary groups, whereas others works efficiently in certain groups. These groups should be avoided in designing a secure cryptographic protocol. For example, the Pohlig-Hellman algorithm can be used to compute discrete logarithms in these groups if $p-1$ is a product of small primes. The index calculus is another method to compute discrete logarithms in these groups, as is the birthday attack (For complete discussion, see [52]). The best algorithm known is called the number field sieve, which is a variation of the index-calculus algorithm. The expected running time of number field sieve algorithm is $L_p[\frac{1}{3}, 1.923]$, where $L_P[x, y] = \exp(y \log p)^x (\log \log p)^{1-x}$, If $x = 1$, the running time is called exponential, while if $x = 0$, the running time becomes polynomials. If $0 < x < 1$ (as in number field algorithm), the running time is called subexponential

(DLP) is fundamental to a number of public-key algorithms, including Diffie-Hellman key agreement [28] and its variants, ElGamal encryption [?], ElGamal signature scheme [?] and its variants and Digital Signature Algorithms (DSA) [1].

A.2 The Diffie-Hellman Problem

The Diffie-Hellman Problem (DHP) is closely related to DLP. It is of significance to public-key cryptography because its apparent intractability forms the basis for the security of many cryptographic schemes. It is well-known that solving DHP can be polynomially reduced to DLP. That is, if DLP can be solved, then obviously the DHP can be easily solved. The converse is still an open problem. DHP can be defined as follows:

Definition A.2 *Given a prime p, a generator α of $\mathbb{Z}_p^*$, and an element α^a mod p and α^b mod p, find α^{ab} mod p.*

The previous definition is the "standard" Diffie-Hellman or as referred in the literature is the *Computational Diffie-Hellman* (CDH) assumption. Under CDH assumption it might well be possible for an adversary to compute something interesting about α^{ab} mod p given α^a mod p and α^b mod p. For example, the adversary might be able to compute the most significant bit, or even half of the bits. This makes the assumption too weak to directly use in typical applications. Another stronger assumption that has been gaining popularity is the *Decisional Diffie-Hellman Problem* (DDH) (For a consistent discussion, see Boneh's survey [15]). It states as follows:

Definition A.3 *Given a prime p, a generator α of $\mathbb{Z}_p^*$, and an element α^a mod p and α^b mod p, distinguish between α^{ab} mod p and any random value α^c mod p.*

A.3 DLP on Elliptic Curves

DLP has many candidates as an underlying algebraic cyclic groups. Set of points on an elliptic curve E which defined over a finite field $\mathbb{F}_q$ is one of these candidates. DLP in this field can be defined as follows:

Definition A.4 *Given an elliptic curve* E *over* $\mathbb{F}_q$, *P a (base) point of* E, *and* $Q \in$ E *find* $x \in \mathbb{Z}$ *such that* $xP = Q$.

Similarly, CDH and DDH can be easily defined on elliptic curves. It is conjectured that DLP on elliptic curves will prove to be more intractable than that in finite fields, the strongest techniques developed for use in finite field do not seem to work on elliptic curves.

Diffie-Hellman key exchange protocol can be described on elliptic curve as following:

1. Alice and Bob agree on an elliptic curve E over a finite field $\mathbb{F}_q$ such that the discrete logarithm problem is hard in $\mathrm{E}(\mathbb{F}_q)$. They also agree on a point $P \in \mathrm{E}(\mathbb{F}_q)$.
2. Alice chooses a secret integer a, computes $P_a = aP$, and sends P_a to Bob.
3. Bob chooses a secret integer b, computes $P_b = bP$, and sends P_b to Alice.
4. Alice computes $aP_b = abP$.
5. Bob computes $bPa = baP$.

A third party knows aP and bP does not be able to calculate abP without solving the discrete logarithm problem. There are some restrictions on choosing the underlying field size q and the representation used for the elements of $\mathbb{F}_q$. Moreover, to avoid some specific attacks, restrictions are placed on the elliptic curves and order of the base point. For more details, see [73].

Appendix B

IBE and Weil Pairing

B.1 Identity Based Encryption:

The idea of identity based encryption was first proposed by Shamir [63]. The purpose was to remove the need for public-key certificates by allowing the user's public key to be a binary sequence corresponding to an information identifying him in a non ambiguous way (e-mail address, IP address combined to a user name, social security number,...). Several practical identity based signature schemes have been proposed over the years [69], but the firstpractical identity based encryption IBE only appeared in 2001 [29]. In this scheme the identity of the user is used as the public key parameter which makes the key management easier and does not require certificates for implementing key revocation. This minimizes overhead of the conventional PKCS and becomes attractive scheme for mobile and resource constrained devices. The identity based encryption is based on four basic algorithms Setup, Extract, Encrypt and Decrypt. The strength of the system is based on Bilinear Diffie-Hellman problem and uses bilinear maps (the Weil or Tate pairing) over supersingular elliptic curves.

B.2 The Weil Pairing:

Pairing of any two points $P, Q \in \mathbb{G}_1$ is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1$ is the additive group and $\mathbb{G}_2$ is the same prime order q The Weil Pairing is defined as:

$$e(P, Q) = \frac{f_P(A_Q)}{f_Q(A_P)}$$

where $f_P(A_Q)$ is the rational function of the divisor with point P of order m evaluated at the divisor with point $Q \in E[m]$, and $f_Q(A_P)$ is the converse. Computing of Weil pairing is the computing of the corresponding rational function of the principal divisor with point P and evaluating it at principal divisor with point Q and vice versa. Miller's algorithm is used for computing the pairing, which is based on the idea of repeated doubling method since the computation involves $1P, 2P, 3P, \cdots, mP$.

B.3 Bilinearity Property

Pairing of points has the bilinearity property, such that:

$$e(2P, P) = e(P+P, P) = e(P, P).e(P, P) = e(P, P)^2 = e(P, P+P) = e(P, 2P)$$

or

$$e(2P, P) = e(P, 2P)$$

also

$$e(aP, bP) = e(P, P)^{ab} = e(abP, P) = e(P, abP)$$

based on the bilinearity; the Identity Based Encryption[2] works as:

$$e(Q_A, P_{pub})^r = e(Q_A, sP)^r = e(Q_A, P)^{rs} = e(sQ_A, rP)$$

The first pairing $e(Q_A, P_{pub})^r$ is used for encryption with public parameters Q_A and P_{pub}. $Q_A = \mathcal{H}(ID_A)$ where ID_A is the identity of the user A (any string of characters). $P_{pub} = sP$ where P is a point on the curve and s is the master key, and r is any random value. The last pairing $e(sQ_A, rP)$ is used for decryption with private key sQ_A and a message parameter rP. We see that due to bilinearity property both encryption and decryption pairings are equal. The security of IBE is based on the Bilinear Diffie-Hellman problem and the Discrete Log problem.

Definition B.1 *Given two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of the same prime order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an admissible bilinear map and let P be the generator of $\mathbb{G}_1$, then Bilinear Diffie-Hellman problem (BDHP) in $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ is to compute $\hat{e}(P, P)^{abc}$ given (P, aP, bP, cP).*

The Decisional Bilinear Diffie-Hellman problem (DBDHP) is, given a tuple of points (P, aP, bP, cP) and an element $h \in \mathbb{G}_2$, to decide whether $h = \hat{e}(P, P)^{abc}$ or not.