Group key agreement in dynamic tactical networks

by

Leila Rahbar

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:
James Davis, Major Professor
Clifford Bergman
Doug Jacobson

Iowa State University

Ames, Iowa

2003

Graduate College
Iowa State University

This is to certify that the master's thesis of

Leila Rahbar

has met the requirements of Iowa State University

Signatures have been redacted for privacy

Table of Contents

ABSTRACT v

1 INTRODUCTION 1
- **1.1 Problem Statement** 2
- **1.2 Organization** 3

2 BACKGROUND 4
- **2.1 Tactical Networks** 4
- **2.2 Key Exchange Protocols** 5
  - 2.2.1 Two party Diffie-Hellman key exchange 5
  - 2.2.2 Initial and auxiliary exchanges 6
  - 2.2.3 Key distribution versus key agreement 6
- **2.3 Layering Authentication** 7

3 DESIGN GOALS 9
- **3.1 Environmental Constraints** 9
- **3.2 Security Goals** 9
- **3.3 Efficiency Goals** 10

4 KEY ESTABLISHMENT IMPLEMENTATIONS 11
- **4.1 Distributory Key Exchange Protocols** 11
- **4.2 Contributory Key Exchange Protocols** 13
  - 4.2.1 The CLIQUES protocol suite 14
  - 4.2.2 Application of CLIQUES in wireless ad hoc networks 20
  - 4.2.3 Tree Group Diffie Hellman 20
  - 4.2.4 AT-GDH Protocol: Initial Key Agreement 23

5 AUXILIARY KEY AGREEMENTS FOR AT-GDH 27
- **5.1 Join Protocols** 27
  - 5.1.1 Single member join 27
  - 5.1.2 Mass join 29
  - 5.1.3 Merge 33
- **5.2 Leave Protocols** 33
- **5.3 Key Refresh** 36

6 ANALYSIS 37
6.1 Security 37
6.2 Efficiency 38
6.2.1 Join 39
6.2.2 Mass join 39
6.2.3 Group merge 40
6.2.4 Member exclusion 41
6.2.5 Key refresh 41

7 CONCLUSION 42

BIBLIOGRAPHY 44

ACKNOWLEDGEMENTS 46

Abstract

Mobile tactical (military) networks have a number of concerns that distinguish them from commercial networks. Of primary concern is information security, achieved in part through message encryption using a common key. These networks are often wireless and ad hoc, that is they lack fixed infrastructure and communications are relayed in a multi-hop fashion. The mobility of the nodes leads to a highly dynamic and unpredictable network topology as well as a dynamic communication group membership.

The focus of this thesis is on finding a secure and efficient solution to group key agreement in a tactical network. Existing group key establishment protocols were surveyed, but many were found inept in this setting. The best solution was the Arbitrary Topology Group Diffie Hellman (AT-GDH). However, this protocol has not been fully specified as no provisions were made for auxiliary key agreements. To complete the AT-GDH key agreement, additional protocols are presented to be performed upon group membership changes. Each protocol was evaluated in terms of efficiency and security. All agreements stemming from additions to the group membership were found to be highly efficient. However, the exponential key structure impedes the efficient removal of one or more participant's contributions.

1 Introduction

Modern tactical or military networks have a number of concerns that distinguish them from commercial networks. Information security has always been a primary concern, which has led to a reliance on encrypted communication. Another concern is survivability, that is the network's ability to perform to some degree after it has been damaged. Furthermore, these networks are to be deployed in a wide range of environments, some of which may be hostile. Finally, as army units are highly mobile, the network must be highly mobile as well. Because of these specific requirements, communication solutions designed for tactical use cannot simply be copies of their commercial counterparts. But it is desirable to make use of civilian technologies as much as possible for the sake of interoperability.

In a typical tactical network scenario, HUMVEE's (military vehicles) form mobile base stations who communicate to each other through wireless or satellite links [LBPH99]. These HUMVEE's then serve as a point of attachment for mobile ground soldiers carrying portable wireless communication devices. If soldiers are outside range of the HUMVEE, they may still communicate by relaying transmissions through other soldiers in a multi-hop fashion. The soldiers do this by forming what is termed an *ad hoc network* amongst themselves. Such a network is highly dynamic in terms of mobility and with nodes frequently joining and leaving. Similar types of ad hoc tactical networks may also be formed among a fleet of ships on the ocean or jets in the air.

For the sake of security, there is a strong need for all wireless communications on the battlefield to be encrypted. This encryption is based upon a shared secret key known by all nodes in the communication group. The process of disseminating the key to all group members is known as *group key establishment*, and may or may not provide a mechanism for creating the key itself.

1.1 Problem Statement

The purpose of this thesis is to develop a secure group key establishment protocol suitable for use in a tactical environment. The nature of the tactical network is assumed to be multi-hop, wireless, highly mobile, and lacking static infrastructures. Because of this non-traditional nature, establishing a group key becomes more challenging than doing so in a wired, static network.

This thesis will contain a description of existing key establishment solutions, and a discussion of their shortcomings in the tactical environment. A key agreement protocol will be proposed by taking an existing initial key agreement protocol and extending it to include auxiliary agreements. These auxiliary agreements include any type of updation of the group key after the initial key has been agreed upon. The addition of the auxiliary agreement protocols should greatly improve the overall efficiency compared to generating a new key from scratch every time a key change is needed. Finally, the efficiency, security, and fault tolerance of the proposed protocol will be analyzed.

The scope of this thesis is limited to the construction and dissemination of the group key. The method for keeping track of the group membership is not discussed. Nor are group membership management operations, such as admitting new members or removing them, discussed. However, an update of the group key may be triggered by a change in group membership. Additionally, it is assumed that the key establishment protocol is to be used in conjunction with some method of authentication. The particular method of authentication is not dwelled upon. Lastly, while a confidential routing protocol is desirable, it is beyond the scope of this thesis.

For the sake of simplicity, it is assumed that all links are two-way and symmetric. That is, all nodes can both transmit and receive messages, and that the radius of transmission is equal to the reception range. In addition, it is assumed that the physical spread of nodes is wider than their transmission ranges. Thus, the transmission of a single node may not immediately reach all members of the network.

1.2 Organization

This thesis is organized as follows. We begin by introducing in Chapter 2 relevant background knowledge of tactical networks as well as the Diffie-Hellman key exchange. Chapter 3 presents our design goals including security and efficiency aspirations. Chapter 4 provides an overview of related work in the area of group key establishment. Chapter 5 contains a description of our own auxiliary agreement protocols. These protocols are then analyzed in Chapter 6 in light of the design goals given in Chapter 3. Finally, conclusions are given and future directions are suggested in Chapter 7.

2 Background

In this chapter, several key concepts are outlined in order to facilitate an understanding of the nature of the tactical environment, the general mechanism of a key exchange, as well as the challenges involved.

2.1 Tactical Networks

Battlefield communication systems supporting command and control functions strongly rely on the use of wireless communication technology. However, unlike cellular networks, mobile tactical networks cannot rely on a fixed network infrastructure. This means there are no static routers, centralized servers, or fixed mobile support stations. The reason for a lack of infrastructure is threefold. First, fixed nodes and central servers are a vulnerable target for an enemy attack. Second, highly mobile military forces need a network that can travel with them. Third, the network needs to stay operative even if some nodes are destroyed or some links become disconnected [SEM99].

Because the multi-hop wireless model does not require a fixed infrastructure and readily adopts to change, it is a good choice of architecture for the dynamic tactical network. Of the types of multi-hop networks, ad hoc networking is the simplest. A mobile ad hoc network (MANET) is basically a self-organized system of peers where transmissions are routed through peer units to other units farther away. However, this type of network may not be entirely accurate for a network of foot soldiers, where most likely there will be mobile base stations with a broader transmission range than the soldier's hand held devices. Thus, transmissions are routed from base-station to base-station before finally arriving to the soldier. This type of network is termed a highly dynamic multi-hop wireless network (HDNet). But for the purpose of simplicity, this thesis will focus on the more basic ad hoc model.

Due to the nature of the tactical network, there are more constraints than typically found in a traditional wired network. Connections are unstable and can be disconnected at any time. In fact, disconnections occur frequently and cannot be easily predicted. This is because

nodes can wander outside of transmission range, run out of battery power, be destroyed by the enemy, or have their communication links jammed. In addition, the network topology can change rapidly as nodes move around, and thus cannot be assumed. Finally, the lack of any central servers imposes an additional challenge on communication protocols.

2.2 Key Exchange Protocols

This chapter provides an overview of the Diffie-Hellman protocol, then explains the difference between different types of key exchanges.

2.2.1 Two party Diffie-Hellman key exchange

The key agreement solution described in this thesis is based upon the Diffie-Hellman key exchange [DH76]. This is a way for two entities to create a shared symmetric key without using a key server. The protocol steps are outlined in Figure 2.1.

1. The parties A and B agree on a cyclic group of order p denoted $\mathbb{Z}_p$, where p is prime, and a generator α for the group. The values p and α are publicly known.
2. A chooses e_A at random, computes α^{e_A} and sends it to B
3. B chooses e_B at random, computes α^{e_B} and sends it to A
4. A computes $K = \left(\alpha^{e_B}\right)^{e_A}$
   B computes $K = \left(\alpha^{e_A}\right)^{e_B}$

Figure 2.1: Diffie-Hellman Key Exchange

The Diffie-Hellman protocol was chosen as a building block for several reasons. First of all, it does not require the use of a third party nor do the two entities need to have a previously shared common secret. Second, the simplicity of the operations is an attractive feature. In addition, the protocol is resistant to passive eavesdropping attacks. Furthermore, since it has been around since 1976, it has been well scrutinized. However, the lack of

authentication between the two parties is a major weakness. Thus, unless a layer of authentication is added on, it can fall prey to the man-in-the-middle attack.

2.2.2 Initial and auxiliary exchanges

When a communication group is first formed, an *initial key agreement* (IKA) is executed in order to provide all members of the group with the encryption key for subsequent communications. Eventually, a member may need to be excluded from the group or a new member may want to join. Furthermore, entire subgroups may need to be excluded at a time or two groups may want to merge. In fact, these membership changes would occur frequently in an ad hoc environment due to frequent disconnections and network partitioning. In order to preserve key independence, the group key must be refreshed after every one of these membership change.

Executing the initial key agreement for every membership change is inefficient as many redundant operations are performed. Thus, it is beneficial to have separate key agreement protocols for each of the single member join, mass join, group fusion, single member leave, and mass leave events. These protocols are collectively termed *auxiliary key agreement* (AKA) protocols. Auxiliary key agreement will be the focus of this thesis.

2.2.3 Key distribution versus key agreement

The methods of group key establishment fall under two main categories, *key distribution* and *key agreement*. Under a distributory protocol, the group key is chosen by a single party and is then passed throughout the group. The emphasis here is on preventing a third party eavesdropper from acquiring the key while it is being distributed. Thus, some type of secure channel is usually needed to keep the key confidential. Since there are no stipulations about the structure of the group key, a problem might arise if the key is chosen unwisely.

In contrast, under a key agreement protocol every member of the group contributes equally towards the key generation. Thus it is also known as contributory key establishment. The participants each generate a secret share and a corresponding public share. They can then

construct the group key by combining their secret share with everyone else's public shares. It is impossible to construct the key without knowing at least one secret share. Since the secret shares are never disclosed, secure channels are not needed. Additionally, each participant is guaranteed that the group key is fresh. However, this type of protocol may be more time consuming since the individual shares need to be given to everyone. It may also require greater computation power as each participant must compute the group key. The Diffie-Hellman key exchange is one example of a contributory protocol.

2.3 Layering Authentication

Since the key exchange protocol itself is not concerned with who the messages originate from, it is necessary to include an authentication layer in order to prevent impersonation. This can be achieved by controlling admissions to the group and performing a group membership verification before every key exchange. Agarwal, et al suggest the use of a "secure group layer" to be interposed between the group communication system and the application [ACTT01]. This layer handles both key agreement and access control. The method of authentication, however, remains as yet an open research topic.

One method for incorporating authentication into group membership management is known as certificate chaining [MAH00], and is designed to operate in the absence of a certification authority (CA). Under this system, nodes cannot be admitted into the group unless they possess a signed certificate. These certificates are issued by a select group of members, known as leaders, who sign the certificate with their own key. The certificate itself contains four items: the identity of the leader, the member's public key, the expiration date, and the leader's signature. The new member can then be considered legitimate by verifying the signature with the leader's public key. Leaders can also instate other leaders in a similar manner, with a separate leadership certificate.

Unfortunately, under this system it is not easy to cancel membership once it is granted. To immediately revoke a member from the group, their identity must be distributed throughout the network. This message may take time to reach all members in a highly volatile ad hoc network, leading to the possibility of the revoked member being accepted as valid. Alternatively, membership can be controlled by choosing a short expiration date for

certificates. Then a member may be revoked by simply not renewing their certificate. The drawback is that leaders must frequently refresh certificates for valid members. Note that if a leader is revoked, all members certified by that leader can no longer be considered valid.

3 Design Goals

This chapter describes the criteria that the key exchange protocol should aspire to meet.

3.1 Environmental Constraints

Environmental constraints of a tactical network include:

- No fixed routing infrastructure exists, including static routers and central servers.
- Nodes are highly mobile, thus no assumptions can be made about the network topology.
- Natural obstructions to wireless signals may exist, leading to frequent disconnections and reconnections.
- Transmission ranges are limited.
- Because it is a military network, the environment may be very hostile.

3.2 Security Goals

Because the environment may be hostile, special emphasis must be placed upon ensuring the security of the key exchange. These security goals are listed and described below.

Group Key Secrecy An outsider should find it computationally infeasible to compute the group key.

Backward and Forward Secrecy Knowledge of the current key should provide no clues in computing old key(s). Similarly, knowledge of old key(s) should provide no clues in computing future key(s). In addition, to prevent excluded members from computing the new group key, this new key should not be distributed using the old group key. Furthermore, newly admitted group members should not have knowledge of any previous group key(s) in use before they joined.

Attack Resistance The protocol should be resistant to active and passive eavesdropping. A *passive eavesdropper* tries to deduce all or part of the group key by listening to traffic on

the network. An *active eavesdropper* not only listens but may also attempt to impersonate a group member, alter messages, or resend old messages. The goal of this type of adversary is to either fool participants into believing they have exchanged a key when they have not, or to cause participants to accept a false key as valid [Sti95]. Furthermore, the protocol should ideally be resistant to denial of service attacks.

Freshness When the group key is updated, it should always be replaced by a fresh key. That is, the new key should be independent of any previous keys, and previous keys should never be reused. This is important to prevent replay attacks and known key attacks.

Key Authentication The group key should only be available to valid group members. Outsiders can only learn of the key through the help of a dishonest participant.

3.3 Efficiency Goals

The protocol should strive to be as simple and efficient as possible, ideally completing before the network becomes disconnected or the topology changes drastically. However, if a node(s) does become disconnected, the protocol should not fail. In addition, any auxiliary exchanges should be at least at efficient as the initial key exchange. In this thesis, efficiency shall be defined in terms of the number of rounds necessary, the number of messages to be sent, the complexity of computations performed at each node, and the total computations performed.

4 Key Establishment Implementations

Much research has been done on securing mobile ad hoc wireless networks, one aspect of which is the encryption of transmissions. Zhou and Haas [ZH99] have proposed a method of securing ad hoc networks through the use of digital signatures and public key encryption. However, this method relies on a subset of nodes to be key servers, and so it is not self-sufficient. Srdjan Capkun, et al [CBH03] have improved the method to be independent of any trusted authority or fixed server and so is more suitable for a peer based network. Unfortunately, neither solution appears appropriate for large scale group communication, which is better achieved through a single symmetric group key rather than multiple public keys.

To facilitate key distribution within a group, several protocols have been proposed. They are of two main categories, key agreement and key distribution, the differences of which were explained in Chapter 2. Most of these protocols were not designed specifically for ad hoc networks, but may be extended to operate in such an environment.

4.1 Distributory Key Exchange Protocols

One key distribution protocol, labeled the centralized key graph protocol [W98], involves a binary tree in which the leaves are the group members, and the nodes are the keys. At the root of the tree is the key server, which constructs the key tree, generates the group key, and distributes the keys to each member via secure channels of communication. These secure channels can be set up, for example, through public key encryption. The intermediate nodes in the tree are intermediate keys, which are also distributed by the key server. Each member knows all keys in their key-path. Thus, if there are eight members in the group, each member knows four keys, including the group key, as illustrated in Figure 4.1.

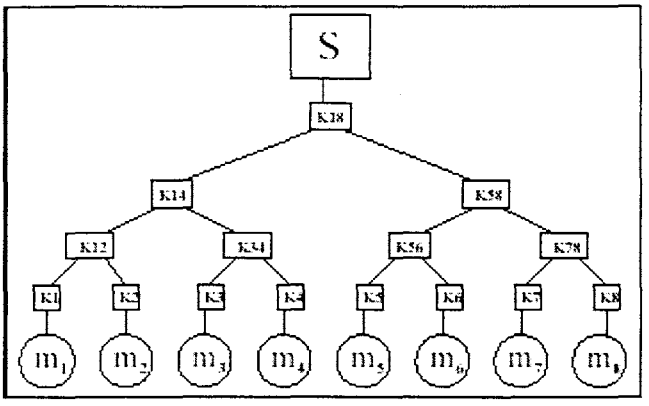

Figure 4.1: Tree structure for a group of 8 members (RBDoo)

Another distributory protocol, labeled optimized rekey [RBDoo], refines the centralized key graph method by eliminating the central key server. Instead, one member is chosen as the group leader, who then constructs the key tree. The key tree is similar to the one shown in Figure 4.1 with all members being leaves of the tree, and the group leader as the left-most leaf. The difference is that the root node is no longer the key server, instead it simply represents the group key.

Referring to Figure 4.1, the protocol works by successively splitting the group into subgroups, until each subgroup contains only two members. At this point, the left-most member (m_1) generates the first mutual key (k_{12}) and sends it to the other member (m_2) via a secure channel. Members m_3, m_5, and m_7 do the same. In the next round, member m_1 generates the second-level key (k_{14}) and sends it to member m_3 via a secure channel. Member m_1 then encrypts k_{14} using k_{12} and multicasts it to m_2. Member m_3 also encrypts k_{14} using k_{34} and multicasts it to m_4. This same process is initiated by member m_5. In the final round, the group leader (m_1) generates the group key (k_{18}) and sends it to the leader of the right-hand branch (m_5). The group leader also encrypts the group key using k_{14} and multicasts the group key to all members of the left-hand branch. The leader of the right-hand subtree (m_5) also encrypts the group key using k_{58} and multicasts it to all members of the right-hand subtree.

So in an eight-membered group, the protocol completes in three rounds and requires seven secure two-party channels. In general, a group consisting of N members will require $\log_2 N$ rounds and N-1 secure channels.

As a candidate for a key exchange solution in the tactical environment, this protocol has many problems. The most difficult is the creation of the key tree in a situation where member's individual transmission ranges are smaller than the overall spread of the group. Thus, it would be difficult to set up a secure channel between two members who are outside of transmission range of each other. Even if these secure channels are possible, they are expensive to set up and maintain as the creation of each channel requires a separate two-party key exchange. Secondly, this protocol relies on multicast communication to distribute the encrypted key. In a wireless tactical environment, this would translate to broadcasting the encrypted key. Again, this could be inefficient if some members lie outside the immediate broadcast range.

Also, this protocol has a disadvantage due to its distributory nature. The generation of the group key by a single member without any one else contributing to it may lead to security issues. For example, that member may choose a previously used key or an inappropriate key which can be guessed by an outsider. There is no guarantee of a fresh key. For this reason, a contributory protocol becomes more appealing.

4.2 Contributory Key Exchange Protocols

One of the simplest two-party contributory key exchange methods is the Diffie-Hellman key exchange. Many solutions for extending the Diffie-Hellman key exchange into a multi-party key agreement have been proposed. The first was done by Ingemarsson et al. [ITW82], in which group members are arranged in a logical ring with each participant adding its exponent to the key before forwarding it to the next participant. Unfortunately, this protocol is unpractical because forming a ring can be difficult in a wireless ad hoc network with limited transmission range. In addition, the symmetry of the protocol inhibits efficient auxiliary key agreements. Furthermore, the protocol can fall prey to passive eavesdropping attacks.

Other existing Diffie-Hellman based key agreement protocols include the Burmester and Desmedt protocol [BD94], the Hypercube protocol [BW98], the Octopus protocol [BW98], and the protocol termed GDH.2 also known as the CLIQUES protocol suite [STW96,

STW98, STW00]. All of these protocols have been found to be lacking in an ad hoc environment [H01]. This is due to either an imposition on network topology, such as the formation of a hypercube, or a reliance on global broadcasts. When comparing the performance of the initial key agreements between these four protocols, the GDH.2 protocol required the least number of both messages and exponential operations [AD02]. In addition, it is resistant to passive eavesdroppers and lays out both the initial key agreement and auxiliary key agreement algorithms. For this reason, the protocol steps are outlined in detail in Section 4.2.1.

An improvement upon GDH.2 is the protocol termed Tree Group Diffie-Hellman (TGDH), which improves efficiency by utilizing a tree structure instead of a linear chain. This protocol is outlined in section 4.2.3. The final protocol discussed in this chapter is the Arbitrary Topology GDH (AT-GDH) protocol. Out of all existing protocols surveyed, it was the only one designed specifically for an ad hoc network. The protocol, which is performed over a spanning tree, is discussed in section 4.2.4.

4.2.1 The CLIQUES protocol suite

The initial key agreement protocol, termed GDH.2, was introduced by Steiner at al. in [STW96] then later extended to include auxiliary agreements in [STW98] and [STW00]. Collectively, the protocols are termed CLIQUES. The CLIQUES protocols are geared towards key agreement in dynamic peer groups, which are generally small-scale groups that utilize a many-to-many communication pattern. Although this approach is not the most efficient nor the best suited for an ad hoc network, it does provide a good starting point as well as a model for auxiliary agreement operations.

The CLIQUES protocols are contributory in nature, meaning each member contributes a share of the key. That share is kept secret, even from other group members. Anyone who has contributed a share can calculate the key, and the key cannot be extracted without knowing at least one of the shares.

Initial Key Agreement (IKA)

The initial key agreement protocol is designed to be executed at the time of group formation. It is assumed that at this time group members do not share any common secrets with each other. The key shares are pieced together in a linear fashion, with the final member broadcasting the intermediate key pieces to all members. The IKA protocol steps are outlined in Figure 4.2.

Initialization: Let n be the size of the group. Let $\mathbb{Z}_p$ be a cyclic group of order q, where p and q are prime, and q is a divisor of $p - 1$. Let α be a generator for the group. Let M_i be the ith member.

Stage 1 (Upflow): Round i ($0 \leq i \leq n$)

1. M_i receives a sequence of (i-1) intermediate key pieces and a cardinal value from M_{i-1}
2. M_i generates a random secret share $N_i \in \mathbb{Z}_q$.
3. M_i computes the new cardinal value $\alpha_i = (\alpha_{i-1})^{N_i}$. The old cardinal value becomes one of the intermediate key pieces.
4. M_i raises the old intermediate key pieces to the power of N_i
5. M_i sends the new cardinal value as well as the new intermediate key pieces to M_{i+1}

$$M_i \rightarrow M_{i+1} : \left\{ \alpha^{\frac{N_1 \ldots N_i}{N_k}} \mid k \in [1,i] \right\}, \alpha^{N_1 \ldots N_i}$$

Stage 2 (Broadcast): Round n

1. M_n generates a random secret share $N_n \in \mathbb{Z}_q$ and computes the intermediate key pieces.
2. M_n broadcasts the (n-1) key pieces to the entire group.

$$M_n \rightarrow \text{ALL } M_i : \left\{ \alpha^{\frac{N_1 \ldots N_n}{N_i}} \mid i \in [1,n] \right\}$$

The final group key is $\alpha^{N_1 \ldots N_n}$

Figure 4.2: CLIQUES IKA protocol

Note that this IKA protocol requires every M_i to perform a total of $(i + 1)$ exponentiations. In order to prevent the computations from increasing as the group size grows, a second IKA protocol has been proposed in [STW00] which utilizes a constant number of exponentiations for each M_i and constant message sizes. This is accomplished by splitting the protocol into four stages, including two broadcast stages. For further details of this protocol, dubbed IKA.2, refer to [STW00].

Auxiliary Key Agreements (AKA)

The auxiliary key agreements are designed to be executed whenever the group membership changes, thereby causing a need for a fresh key. The auxiliary protocols build upon the partial keys broadcast in stage 2 of IKA, thus any member with knowledge of the partial keys can initiate an AKA. This initiator is termed the group controller, denoted by M_c. The types of AKA's are: member addition, member exclusion, mass join, subgroup exclusion, group fusion, and key refresh.

In the member addition protocol, the group controller modifies its secret share in order to prevent the new member from learning the old group key. It then sends an upflow message to the new member containing the intermediate key pieces. Once the new member receives the message from the group controller, it adds its own secret share to it and broadcasts the new key pieces to all members of the group. This protocol is outlined in Figure 4.3.

Stage 1 (Upflow): round 1

1. M_c generates a new exponent $\hat{N}_c$ and changes its secret share from N_c to $\hat{N}_c N_c$, then computes the new key pieces.
2. M_c sends the intermediate key pieces and the cardinal value to the new member.

$$M_c \rightarrow M_{n+1} : \left\{ \alpha^{N_c \Pi\left(N_p \mid p \in [1,n] \wedge p \neq k\right)} \mid k \in [1,n] \right\}, \alpha^{N_c \Pi\left(N_p \mid p \in [1,n]\right)}$$

Stage 2 (Broadcast): round 2

1. The new member generates a new secret share N_{n+1} and computes the key pieces.
2. The new member broadcasts the key pieces to all members of the group

$$M_{n+1} \rightarrow \text{ALL } M_i : \left\{ \alpha^{N_c \Pi\left(N_p \mid p \in [1,n+1] \wedge p \neq k\right)} \mid k \in [1,n+1] \right\}$$

Figure 4.3: CLIQUES Member Addition

A slightly different protocol was designed by Steiner at al. to handle mass group additions, Figure 4.4. This is preferable to repeatedly running the single member addition operation as only one broadcast is performed instead of a separate broadcast for each new member.

Initialization: Let m be the number of new group members.

1. M_c generates a new exponent $\hat{N}_c$ and changes its secret share from N_c to $\hat{N}_c N_c$, then computes the new key pieces.
2. M_c sends the intermediate key pieces and the cardinal value to the first new member.

$$M_c \rightarrow M_{n+1}: \left\{ \alpha^{N_c \Pi\left(N_p \mid p \in [1,n] \wedge p \neq k\right)} \mid k \in [1,n] \right\}, \alpha^{N_c \Pi\left(N_p \mid p \in [1,n]\right)}$$

Stage 1 (Upflow): round i ($1 \leq i \leq m$)

1. The ith new member generates a secret exponent N_{n+i} and computes the new key pieces
2. The ith new member sends the intermediate key pieces and the cardinal value to the next new member.

$$M_{n+i} \rightarrow M_{n+i+1}: \left\{ \alpha^{N_c \Pi\left(N_p \mid p \in [1,n+i] \wedge p \neq k\right)} \mid k \in [1,n+i] \right\}, \alpha^{N_c \Pi\left(N_p \mid p \in [1,n+i]\right)}$$

Stage 2 (Broadcast): round m+1

1. The final new member generates a new secret share N_{n+m} and computes the key pieces.
2. The final new member broadcasts the key pieces to all members of the group

$$M_{n+m} \rightarrow \text{ALL } M_i: \left\{ \alpha^{N_c \Pi\left(N_p \mid p \in [1,n+m] \wedge p \neq k\right)} \mid k \in [1,n+m] \right\}$$

Figure 4.4: CLIQUES Mass Join

Another membership change event occurs when two distinct groups merge into a single group; this is referred to as group fusion. One approach to group fusion is based on a technique developed by Steer at al. in [BW98]. Under this approach, the group controller from each group exchanges their key residues: α^{K_1} and α^{K_2} respectively, and computes the new group key $K_{12} = \alpha^{K_1 K_2}$. They then broadcast the new key pieces to all members in the group. If there arises a need to revert back to the two original groups, group fission, the two groups simply use their original keys (K_1 and K_2) again.

The third auxiliary operation is member exclusion, Figure 4.5. In this case it is necessary to change at least one remaining member's share to prevent the excluded member from obtaining the new key. In this solution, it is the sponsor who changes its secret share.

Initialization: M_c generates a new exponent $\hat{N}_c$ and changes its secret share from N_c to $\hat{N}_c N_c$, then computes the new key pieces.

Stage 1 (Broadcast): round 1

The group controller broadcasts the new key pieces to all members of the group except the recently excluded member, d.

$$M_c \rightarrow \text{ALL } M_i : \left\{ \alpha^{\Pi\left(N_p \mid p \in [1,n] \wedge p \neq k\right)} \mid k \in [1,n] \wedge k \neq d \right\}$$

Figure 4.5: CLIQUES Member Exclusion

The authors extend the member exclusion protocol to handle the removal of subgroups by a simple modification to the broadcast message. Only the key pieces correlating to the remaining members are distributed.

The final auxiliary protocol describes the steps to refresh the group key, Figure 4.6. Unlike the other protocols, this one does not result from a membership change event, but should be performed periodically for greater security.

Initialization: Let M_h represent the member initiating the key-refresh. This member changes its secret exponent from N_h to $\hat{N}_c N_c$ and computes the new key pieces.

Stage 1 (Broadcast): round 1

The initiator broadcasts the key pieces to all members of the group

$$M_h \rightarrow \text{ALL } M_i : \left\{ \alpha^{N_h \Pi\left(N_p \mid p \in [1,n] \wedge p \neq k\right)} \mid k \in [1,n] \right\}$$

Figure 4.6: CLIQUES Key Refresh

4.2.2 Application of CLIQUES in wireless ad hoc networks

The CLIQUES protocols are an improvement over the Ingemarsson et al. and Burmester and Desmedt protocols because fewer exponential operations are needed. However, there arise three main issues with respect to the wireless ad hoc environment. First, the global broadcast performed in the last step of each protocol may not be feasible in a network where some members lie outside the transmission range of member M_n. Second, the linearity of the steps can lead to problems in an environment where frequentdisconnections are inevitable. For example, if a member node goes offline before it has a chance to send the key pieces to the next member in the sequence, then the protocol cannot complete until that node resumes connectivity. Third, the protocol does not provide a method for determining the sequence of the linear chain. Such a sequence cannot be predetermined, as nodes are highly mobile, and so must be somehow integrated into the protocol.

Anton and Duarte have proposed a method for group key establishment in wireless ad hoc networks by solving the problem of sequence establishment [ADO2]. This is done through a group member discovery protocol to be used during the execution of the CLIQUES protocols. When a node is ready to send the key pieces to the next node in the chain, it searches for nearby group members through the use of flooding messages. Every node that receives the flooding message and that has not yet participated sends a reply message. The searching node then picks one to be the next node in the sequence. If no reply messages are received, the node assumes it is the last in the chain and performs the final step of the group key establishment.

While Anton and Duarte's solution may be viable in the ad hoc environment, it still does not solve the issues related to the linearity of the sequence or the global broadcasts.

4.2.3 Tree Group Diffie Hellman

Another group key agreement protocol proposed by Kim, Perrig, and Tsudik is known as tree group Diffie Hellman (TGDH) [KPTOO], [KTPO2]. This protocol is based on the Diffie-Hellman key exchange and makes use of binary key trees for efficient key computation. The key tree has the same structure as the tree in the optimized rekey protocol (Figure 4.1), with

the leaves of the tree representing group members' session keys, the root node representing the group key, and intermediate nodes representing intermediate keys. The authors use the notation $<l,v>$ to denote each node in the tree, meaning the v-th node at level of depth l. A sample tree for a group consisting of six members is shown in Figure 4.7.

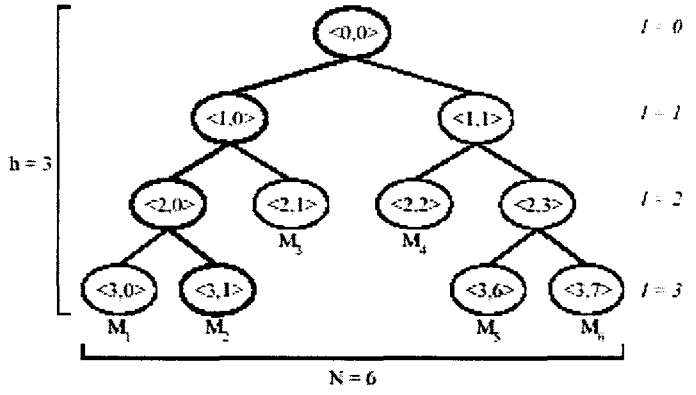

Figure 4.7: TGDH tree with notation (KPT00)

Each node is associated with both a key and a blinded key. The blinded key is computed using a modular exponentiation equivalent to the Diffie-Hellman protocol. Every member knows every key in its key-path from itself to the root, as well as the blinded keys of every node in the tree. The group key is computed recursively starting with the highest level and working up to the root. Each key is computed individually from the blinded key of the right child raised to the exponent of the key of the left child as follows:

$$K\langle l,v\rangle = \left(BK\langle l+1,2v+1\rangle\right)^{K\langle l+1,2v\rangle} \bmod p$$

Or alternatively, the key can be computed from the blinded key of the left child raised to the exponent of the key of the right child. The blinded key itself is calculated as follows:

$$BK\langle l,v\rangle = \alpha^{K\langle l,v\rangle} \bmod p$$

Thus, any member can compute the group key $K_{<0,0>}$ from that member's secret share and from the keys in each members key-path, which are in turn derived from the blinded keys of the siblings of each node in the key-path. This set of blinded keys is termed by the authors the *co-path*. Additionally, the authors note that the group secret $K_{<0,0>}$ is not the final group key. Instead, a hash function $h(K_{<0,0>})$ is used for the purposes of authentication, encryption,

and integrity. For the purposes of authentication, all protocol messages are signed by the sender.

The authors also provide the mechanisms for updating the group key upon a membership change event. These events include join, leave, merge, partition, and key refresh. In each case, the protocol is initiated by a member termed the sponsor.

Under the join protocol, the key-tree is modified to include a leaf representing the new member. This leaf is added in such a way as to minimize the height of the tree. The sponsor member collects the new member's blinded key and uses it along with every current members' blinded keys to compute the new group key. The sponsor then broadcasts the new tree as well as all blinded keys. Once the current members receive the new tree structure as well as the blinded keys, they can compute the new group key.

When a member leaves the group, the sponsor deletes the corresponding leaf from the tree, then restructures the tree to minimize its height. The sponsor member then changes its own secret share, thereby changing its blinded key, and computes the new group key. Finally, the new tree structure and set of blinded keys are broadcast to all group members.

The key refresh protocol is a special case of the leave protocol, but without the deletion of any nodes. Instead, the sponsor member simply changes its secret share.

If it becomes necessary to partition the group or remove multiple members at once, those leaving members are first deleted from the tree. Each leaving member is associated with a non-leaving sponsor. The right-most sponsor changes its secret share. Then each sponsor broadcasts the set of blinded keys on its key-path.

Due to network faults, separate subgroups can form which may need to be merged once the network heals. In this case, a sponsor from each tree broadcasts its tree structure along with all blinded keys to the other group. Assuming there are only two merging groups, the shallower tree is joined to the deeper tree. A third sponsor is chosen who then broadcasts the new key structure and all blinded keys. At this point all members can now compute the new group key.

This protocol has a major advantage over GDH.2 of the Cliques suite as the tree structure leads to more efficient key computation. More specifically, the number of exponentiations

required for member addition under TGDH is $O(\log_2 n)$ compared with $O(n)$ for GDH.2. However, it still relies on global broadcast messages. Furthermore, the basic tree structure is preserved throughout all key updates. This not only does not take into account member movements, which occur frequently in a highly mobile tactical network, but it also means that any sponsor needs to keep track of the tree structure.

4.2.4 AT-GDH Protocol: Initial Key Agreement

Solving the problem of the global broadcasting issues, another protocol, based on the construction of spanning trees, was proposed by Hietalahti in [Hie01]. The author calls this protocol AT-GDH, for arbitrary topology group Diffie-Hellman. This protocol was designed specifically for use in mobile ad hoc wireless networks, where connections may be unreliable, broadcast range is limited, and nothing can be assumed about the topology of the connection graph. Thus it is the most appealing choice for use in a tactical environment out of all of the existing solutions.

Similar to TGDH, AT-GDH also uses a hierarchical tree structure for computing the group key, with the root node representing the group key. However, this tree is not necessarily binary, and instead of only the leaves representing group members, every node in the tree represents a member of the group. Since the tree is a subgraph of the connectivity matrix, and since the set of vertices equals the set of group members, it is termed a spanning tree.

The author provides one possible method for constructing the spanning tree, assuming that the nodes are aware of their neighbors. It begins with the initiator sending a message to each of its neighbors. Once the neighbors get the message, they send back an acknowledgement. At this point, the initiator becomes the root of the tree and each neighbor becomes a child of the root. They then send a message to their neighbors, until every node is part of the tree. If a node receives more than one message, it only acknowledges the first and ignores the others. However, if the network is extremely unreliable it may be necessary to send a negative acknowledgement instead of merely ignoring the message. A node assumes that it must be a leaf in the tree if it does not receive any acknowledgements at all. Note that the nodes are not aware of the entire tree structure, they only know of their parent and children (if any).

A sample network connectivity matrix along with a possible corresponding spanning tree is given in Figure 4.8. In this case, the initiator is the node A.

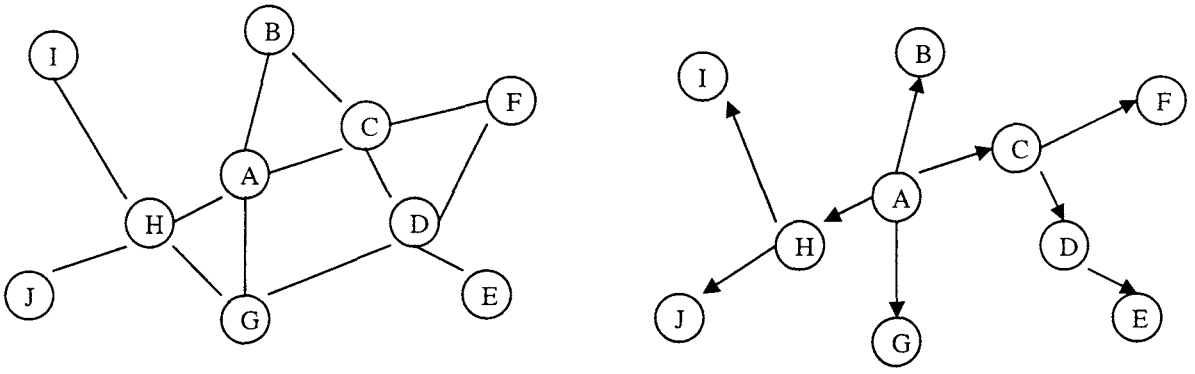

Figure 4.8: Network connectivity matrix and corresponding spanning tree

Once the spanning tree has formed, the initial key agreement can begin. The protocol begins with the leaves of the tree and culminates with the final group key at the root node. The root node then passes the key pieces down to its children, which pass them to their children, until everyone has the necessary information to construct the group key. Thus, every node sends one message and receives one message. Since a node may have more than one child, it first forms a Diffie-Hellman key with its left-most child. Then it uses this new key and forms another Diffie-Hellman key with the second child. The process continues for all remaining children. The key formed with the last child serves as the exponent to be passed to the parent. In the second stage, the root sends a message to its children containing the public keys of the right-hand siblings and the recursive exponentiations of the left-hand siblings. This propagates down to the leaves. The protocol steps are given in Figure 4.9.

The tree structure of AT-GDH makes it more efficient than the linear-based GDH.2 of the Cliques suite, as there are less exponentiations performed at each node and less rounds. AT-GDH requires $O(\log_k n)$ exponentiations and the same number of simple rounds, compared to O(n) exponentiatons and rounds of GDH.2.

While TGDH is of a comparable efficiency to AT-GDH ($O(\log_2 n)$ compared to $O(\log_k n)$), AT-GDH has other advantages over it. As mentioned earlier, AT-GDH does not require a global broadcast to disseminate the key. In addition, nodes in the spanning tree of AT-GDH are at most one hop away from each other in the ad hoc network. This is not necessarily the case for the binary tree of TGDH. Thirdly, the entire tree structure does not have to be

propagated throughout the group, as it is sufficient to simply know the immediate parent and children nodes.

One drawback of the AT-GDH protocol is the reliance on a bijection, termed φ, from a cyclic group G to the set of integers modulo q. This is the same bijection used in the Hypercube and Octopus protocols, both of which were presented by Becker and Wille. The need for this bijection stems from the use of a Diffie-Hellman key (which is an element of G) to be used as an input in the next round. If alpha is chosen to be the primitive element modulo p, in other words if $G = \mathbb{Z}_q^*$ then a solution for φ is simple. But otherwise, a practical solution for φ has yet to be found. Becker and Wille themselves acknowledge this problem in [BW98].

Initialization: Let G be a cyclic group of order q with generator α. Let h represent the height of the spanning tree. Assume there exists a bijection $\varphi: G \rightarrow \mathbb{Z}_q$

Stage 1:

Round 1

1. All leaves of the tree select a random secret share $k_x \in \mathbb{Z}_q$
2. The leaves send the blinded share to their parents

   $x \rightarrow y : \alpha^{k_x}$

Rounds 2..h: For all nodes x that have children:

1. Node x selects a random secret exponent $e_x \in G$
2. Node x waits for the blinded shares $\alpha^{k_{x.j}}$ from all of its children
3. Node x calculates $k_x = \varphi(K(x,c_x))$ from

   $K(x,0) = e_x$

   $K(x, j) = \alpha^{k_{x.j}\varphi(K(x.j-1))}$ for $1 \leq j \leq c_x$
4. $x \rightarrow y : \alpha^{k_x}$

Stage 2:

Round $h + 1$

1. The root node, ε, sends a message to each of its children containing the key pieces

   $\varepsilon \rightarrow x.i : Mx.i = \left\langle -, \alpha^{\varphi(K(x,i-1))}, \alpha^{k_{x.(i+1)}}, \alpha^{k_{x.(i+2)}}, \alpha^{k_{x.cx)}} \right\rangle$

Rounds $h + l$ where $2 \leq l \leq h$

1. Node x sends a message to each of its children containing the key pieces

   $x \rightarrow x.i : M_{x.i} = \left\langle M_x, \alpha^{\varphi(K(x,i-1))}, \alpha^{k_{x.(i+1)}}, \alpha^{k_{x.(i+2)}}, \alpha^{k_{x.cx)}} \right\rangle$

The final group key is $k_\varepsilon = K(\varepsilon, c_\varepsilon)$

Figure 4.9: Spanning Tree IKA

5 Auxiliary Key Agreements for AT-GDH

Comparison of the group key exchange protocols given in Chapter 4 shows the AT-GDH protocol to be best suited for use in a dynamic wireless ad hoc environment. However, the initial key agreement protocol described in [Hie01] forms only part of a key establishment system. Once a key has been in use for some time, it is likely that the communication group will want to change to a new key. Reasons for refreshing the key could include the current key being compromised or a change in group membership. Changes to group membership, such as additions and exclusions, occur quite frequently in practice, especially in an ad hoc environment where there are frequent disconnections and network partitioning. Although it would suffice to re-execute the initial key agreement protocol every time the key needs refreshed, more efficient alternatives are desirable. These alternatives are collectively termed auxiliary key agreements.

In this chapter, several different auxiliary key agreement algorithms are described. The algorithms are join, mass join, merge, leave, and refresh. Each has been derived through modification of the initial algorithm. The result in all cases is a completely new group encryption key.

5.1 Join Protocols

There are three different ways a communication group can be expanded: the addition of a single member, the addition of multiple members, or the addition of an existing group. Each case requires a slightly different key agreement algorithm. These algorithms are described below.

5.1.1 Single member join

In the case of a single member being added to the group, two things are required. The first requirement is that a new group encryption key needs to be formed from contributions of all

members including the newly joined member. The second is that this key needs to be securely propagated throughout the group.

These requirements can be achieved through a simplification of the initial key agreement algorithm. In this new algorithm, the joining member chooses its own secret share, call it e_j. Then the new key is simply $\left(\alpha^{old\ key}\right)^{e_j}$. To compute this new key, the new member needs to know $\alpha^{(old\ key)}$ and all original members need to know α^{e_j}. Thus the secrecy of the old key is not disclosed to the new member. Furthermore, it is resistant to passive eavesdropping since the knowledge of both $\alpha^{(old\ key)}$ and α^{e_j} does not imply knowledge of the old key, e_j, or the new key formed.

The single member join algorithm is given in Figure 5.1 below.

Phase 1, Round 1

1. Sponsor, *a*, agrees to add new member, *j*, to the group.
2. New member, *j*, chooses a secret key, e_j, and sends the blinded version to the sponsor.

   $$j \rightarrow a : \alpha^{\varphi(e_j)}$$

3. $a \rightarrow j$: $\alpha^{\varphi(old\ key)}$

Phase 2

Round 2: The sponsor sends the new member's share to all nodes within range.

$$a \rightarrow c_a : \alpha^{\varphi(e_j)}$$

Rounds 3 to 3+*l*, $l = 1,\ldots,h$: For every node *x* on level *l*,

$$x \rightarrow c_x : \alpha^{\varphi(e_j)}$$

The resulting group key is:

$$\text{key} = \left(\alpha^{\varphi(e_j)}\right)^{\varphi(old\ key)} = \left(\alpha^{\varphi(old\ key)}\right)^{\varphi(e_j)}$$

Figure 5.1: Single member join

5.1.2 Mass join

A mass join event occurs when multiple new members are admitted into an existing group simultaneously. In this case, a new group key must be agreed upon and issued to each new and existing member.

This can be done using a strategy similar to the star topology based protocol given in [SSDW88]. All new members are admitted through the aid of a single sponsor member, who forms the center of the star. This sponsor collects the contributions (α^{e_k}) of each new member, compiles them together, then computes the new group key. The sponsor then distributes the new members' contributions throughout the group allowing all previously existing members to compute the new group key. Finally, for the new members to compute the group key, they need to know $\alpha^{(old\ key)}$ as well as the contributions of their fellow new members. Note that for this protocol to work effectively, all joining members should be within transmission range of the sponsor member.

For example, if three new members are added, the sponsor would collect the following shares:

New member 1 → sponsor: α^{k_1}
New member 2 → sponsor: α^{k_2}
New member 3 → sponsor: α^{k_3}

Next, the sponsor would compute the group key as follows:

Sponsor selects a random e_s
contributions = $\alpha^{k_3\alpha^{k_2\alpha^{k_1 e_s}}} = \alpha^{k_3} \wedge \left(\alpha^{k_2} \wedge \left(\alpha^{k_1 e_s}\right)\right)$
new group key = $(\text{contributions})^{(\text{old key})}$

Once each existing member receives the compiled new member's contributions from the sponsor, they can also compute the new group key. Finally, the sponsor sends the following key pieces to the new members, allowing them to compute the group key:

Sponsor → new member 1: $\left\langle \alpha^{(\text{old key})}, \alpha^{e_s}, \alpha^{k_2}, \alpha^{k_3} \right\rangle$
Sponsor → new member 2: $\left\langle \alpha^{(\text{old key})}, \alpha^{k_1 e_s}, \alpha^{k_3} \right\rangle$
Sponsor → new member 3: $\left\langle \alpha^{(\text{old key})}, \alpha^{k_2} \wedge \left(\alpha^{k_1 e_s}\right) \right\rangle$
Sponsor → existing members: $\alpha^{(contributions)}$

Each new member would compute the group key as follows:

New member 1 computes: key = $\alpha^{(old\,key)} \wedge \left(\alpha^{k_3} \wedge \left(\alpha^{k_2} \wedge \left(m_1\right)^{k_1}\right)\right)$ where $m_1 = \alpha^{e_s}$

New member 2 computes: key = $\alpha^{(old\,key)} \wedge \left(\alpha^{k_3} \wedge \left(m_2\right)^{k_2}\right)$ where $m_2 = \alpha^{k_1 e_s}$

New member 3 computes key = $\alpha^{(old\,key)} \wedge \left(m_3\right)^{k_3}$ where $m_3 = \alpha^{k_2} \wedge \left(\alpha^{k_1 e_s}\right)$

Each existing member would compute the group key as follows:

key = $\left(m_x\right)^{(old\,key)}$ where $m_x = \alpha^{(contributions)}$

The mass join algorithm is given in Figure 5.2 below.

Round 1

1. Sponsor, *s*, agrees to add *n* new members to the group.
2. *s* selects a random $e_s \in G$
3. Each new member, *j*, chooses a secret key (k_j), and sends their contribution to *s*.

   For *j* in (1..*n*):

   $$j \rightarrow s : \alpha^{k_j}$$

Round 2

1. *s* waits to receive α^{k_j} for all $j = 1,..,n$
2. *s* calculates $contributions = \varphi(K(n))$ from

   $$K(0) = e_s$$
   $$K(j) = \left(\alpha^{k_j}\right)^{\varphi(K(j-1))} \quad \text{for } j = 1,...,n$$

3. $s \rightarrow j : M_j$, where

   $$M_j = \left\langle \alpha^{\varphi(oldkey)}, \alpha^{\varphi(K(j-1))}, \alpha^{k_{j+1}}, \ldots, \alpha^{k_n} \right\rangle$$

4. $s \rightarrow x : \alpha^{(contributions)}$ where *x* are all existing members

The resulting group key is: $\alpha^{(contributions)\varphi(oldkey)}$

Figure 5.2: Mass join

This protocol can be further optimized if the joining members together with the sponsor member formed a spanning tree among themselves, depicted in Figure 5.3, rather than a star topology. This would not only distribute the computations better, but is a more effective solution if not all joining members are within transmission range of the sponsor. However, at least one joining member must be within transmission range of the sponsor, and all joining members must form a connected graph. The optimized protocol is given in Figure 5.4.

Note that neither version of the mass join protocols require the formation of a spanning tree among existing members.

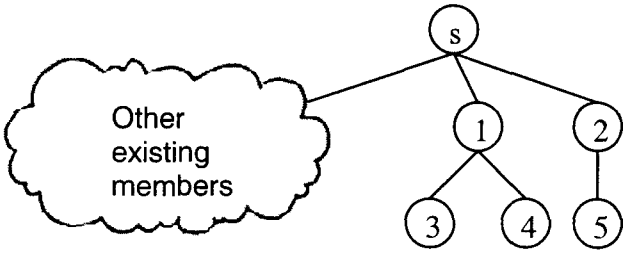

Figure 5.3: Mass addition of 5 new members in tree form

Round 1

1. Sponsor, s, agrees to add n new members to the group.
2. s initiates the formation of a spanning tree of height h consisting entirely of new members with itself as the root.
3. For all nodes $v = y.i$ with $c_v = 0$ (the leaves of the tree):
   a. v selects a random $k_v \in \mathbb{Z}_q$
   b. $v \rightarrow y$: α^{k_v}

Rounds 2 ... h-1

See IKA protocol, rounds 2 ... h

Round h

1. s selects a random $e_s \in G$
2. s waits to receive α^{k_j} for all $j = 1,..,c_s$
3. s calculates *contributions* = $\varphi(K(c_s))$ from

$$K(0) = e_s$$
$$K(j) = \left(\alpha^{k_j}\right)^{\varphi(K(j-1))} \quad \text{for } j = 1,\ldots,c_s$$

4. for i in $(1..c_s)$:

   $s \rightarrow i$: M_i where

$$M_i = \left\langle \alpha^{\varphi(oldkey)}, \alpha^{\varphi(K(i-1))}, \alpha^{k_{i+1}}, \ldots, \alpha^{k_{c_s}} \right\rangle$$

5. $s \rightarrow x : \alpha^{(contributions)}$ where x are all existing members

Rounds $h+l$

See IKA protocol, rounds $h+l$

The resulting group key is: $\alpha^{(contributions)\varphi(oldkey)}$

Figure 5.4: Optimized mass join

5.1.3 Merge

A merge occurs when an entire group joins another group to form a super-group. This is different than the mass join event, as the individuals joining have a pre-existing relationship with each other.

One approach is to apply the mass join protocol given in section 5.1.2 to the smaller group. Another approach to group merge has been introduced in [BW98] then further discussed in [STW00]. Under this approach, the two groups exchange their key residues: α^{K_1} and α^{K_2}, where K_1 and K_2 are the respective group keys. The super-group key now becomes $\alpha^{K_1K_2}$. Not only is this method quick and simple, but it also allows for the super-group to rollback to the original two groups by switching they key back to either K_1 or K_2.

This approach may be extended to allow the merge of three or more groups. This can be done by designating one member of each group as group controller. The group controllers would then form the nodes of a spanning tree. The super-group key could then be formed by performing the AT-GDH initial key agreement using each node's own group key as their secret key. The final super-group key would then take the form:

$$\alpha^{K_n\alpha^{K_{n-1}\alpha^{\cdots^{K_3\alpha^{K_1K_2}}}}}$$

where $K_1,\ldots,K_n$ are the individual group keys. One stipulation of this method is that each group controller must form a connected graph among themselves.

5.2 Leave Protocols

When a member leaves the group, it is desirable to form a new group key in order to exclude that individual from subsequent communications. It is assumed that upon the removal of that individual, all other nodes remain connected.

According to the cliques protocol suite, this is done by simply changing the secret share of one of the remaining members, then broadcasting the set of all subkeys minus the newly excluded member's subkey. Thus the newly excluded member is unable to compute the new group key.

Unfortunately, this procedure fails when applied to AT-GDH. The reason being the key structure of AT-GDH is vastly different than the key structure of the cliques protocol suite.

AT-GDH key structure: $\alpha^{N_n} \wedge \left(\alpha^{N_{n-1}} \wedge \ldots \left(\alpha^{N_3} \wedge \left(\alpha^{N_1 N_2}\right)\right)\right)$

Cliques key structure: $\alpha^{N_1 N_2 N_3 \ldots N_n}$

For example, suppose the root of the spanning tree took the following steps to come up with a new group key. First it would choose a new secret share, k_{new}. Then it would compute the new group key as $(\alpha^{knew})^{\varphi(old\ key)}$. Finally it would disseminate α^{knew} to all members throughout the spanning tree except the newly excluded member. However, if the newly excluded member was to eavesdrop on the communication, it could easily discern the value of α^{knew}. Since this excluded member already knows the value of φ(old key), it can easily compute the new group key. Therefore, this is not a desirable method of choosing a new group key.

In order for the excluded member not to be able to compute the new group key, that member's share must be removed from the key computation. This removal is fairly easy when the Cliques key structure is used, but much more complicated under the AT-GDH key structure. For example, referring to Figure 5.5, member 3 is to be excluded from the group. Assume for now that the spanning tree structure has not changed since the initial key exchange. Then the root node, member 1, chooses a new secret share, k_{new}, and comes up with a new group key as follows:

New group key = $\alpha^{N_2} \wedge \left(\alpha^{N_4 k_{new}}\right)$

Where $N_2 = \alpha^{N_5} \wedge \left(\alpha^{N_6 e_2}\right)$ and $N_4 = \alpha^{N_7 e_4}$

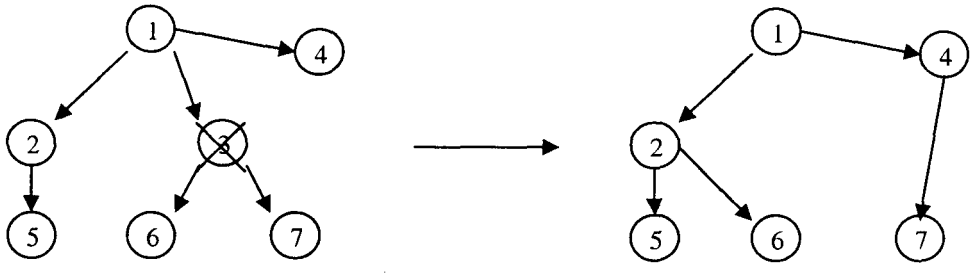

Figure 5.5: Removal of member 3

This requires a re-calculation of the shares N_2 and N_4. Unfortunately, the root node does not possess the knowledge required to calculate N_2 and N_4. Namely, it does not know N_5, N_6, N_7, e_2, or e_4. Thus this information needs to be passed from the leaves upward to the root in a manner similar to phase 1 of the initial key agreement protocol. Finally, the key pieces can be passed downwards in the same way as phase 2 of the initial key agreement protocol.

More realistically, it does not make sense to re-use the same key tree as was used for the initial key agreement due to the mobility of each individual member. For example, a member who was the direct child of the root node during the initial exchange may no longer be within immediate transmission range during the member exclusion operation. Additionally, the initiator of the initial key agreement, and the initiator of the member exclusion operation may not be the same node. Thus the root node may represent a different member in each case.

Because of the need to form a new spanning tree combined with the need to perform both phase 1 and phase 2 of the initial key agreement, the member exclusion protocol becomes equivalent in efficiency to IKA. The only difference is that each node in the spanning tree (except the root node) does not have to select a new random secret share, they may reuse their original value.

The mass exclusion operation faces the same problem as the single member exclusion operation, that is it is difficult to remove the excluded members' contributions from the group key. Thus, in order to prevent the excluded members from acquiring the new group key, a new spanning tree should be formed sans the excluded members. Then the IKA protocol can be performed on this new spanning tree to yield a new group key.

Refreshing the key after a group division follows a similar pattern. To split a single group into n disjoint groups, one member from each group should be chosen to initiate the protocol. Each chosen initiator then forms a spanning tree consisting entirely of their own members, with themself as the root. The IKA protocol can then be performed on each spanning tree resulting in n new group keys.

5.3 Key Refresh

The final auxiliary operation to be described is the key refresh. Depending on policy, the key may need refreshing for reasons other than a change in group membership. For example, keys can be set to expire after a certain length of time, forcing the key to be periodically refreshed. The group will also want to refresh the group key upon the disclosure of either the group key itself or a member's secret key. Because of the risk of a member's secret key being compromised, it makes sense to periodically refresh individual member's contributions as well as the group key.

The actual protocol turns out to be even simpler than the single-member join case. The member whose contribution requires refreshing, i.e. its contribution is the oldest, initiates the protocol by generating a new secret key k_{new}. This member then distributes its new contribution $\alpha^{k_{new}}$ to all group members. The new group key becomes $\left(\alpha^{k_{new}}\right)^{\varphi(oldkey)}$. Note that this new group key is independent from the previous group key.

6 Analysis

This chapter presents an analysis of the solution based on the design goals given in Chapter 3.

6.1 Security

Group Key Secrecy All of the protocols described in Chapter 5 are an extension of the two party Diffie-Hellman key exchange. Thus, the security of the protocols rest upon the intractability of the Diffie-Hellman problem. That is, given $\alpha^{k_1} \bmod p$ and $\alpha^{k_2} \bmod p$, it is infeasible to compute $\alpha^{k_1 k_2} \bmod p$. Since the mass join and exclusion protocols depend heavily upon the IKA protocol, they are at most as secure as the IKA. We leave it as an unproven conjecture that all of the protocols given in Chapter 5 can be reduced to the Diffie-Hellman problem.

Backward and Forward Secrecy We claim that excluded members cannot compute the new group key and newly admitted members cannot compute the previous group key. When a group member is excluded, the key structure is reconstructed in such a way that it does not include the excluded member's contributions. In addition, this new group key is never encrypted using a previous key. So the excluded member is prevented from discovering the new key. On the other hand, when a new member is admitted, they are given the value $\alpha^{(old\ key)} \bmod p$. Computing the old key from this value is equivalent to solving the discrete logarithm problem.

Attack Resistance If the key agreement protocols are coupled with an authentication scheme, a non-member should not be able to fool participants by impersonation, altering messages, or by resending old messages. However, the protocols may be vulnerable to denial of service (DoS) attacks. If a parent node is not able to send messages, none of its children will be able to receive the key.

Freshness Since every method of updating the key given in Chapter 5 requires at least one Diffie-Hellman key exchange with a new or different contribution, this guarantees a fresh group key that has never been used before.

6.2 Efficiency

In this section, the complexity of each auxiliary protocol is described, and compared to the complexity of the initial key agreement. For a discussion of the efficiency of the initial key agreement, refer to [Hie01]. The efficiency measures include the number of messages sent, the number of simple rounds, the number of synchronous rounds, the local computation, and the total computation. In a simple round, each node can only send and receive one message. But in a synchronous round, nodes can send and receive many messages. The local computation is defined as the maximum exponentiations each node is expected to compute. The total computation is then the sum of local computations over all nodes. Efficiency findings are summarized in Table 6.1 and Table 6.2.

For the purpose of estimating efficiency measures, it is assumed that the spanning tree is k-ary (each node has k children) and perfectly balanced. Let n represent the total number of nodes, j represent the number of joining nodes, and h represent the height of the tree.

Table 6.1: Efficiency - messages and rounds

	Messages	Simple Rounds	Syn. Rounds
IKA	$2n$-2	$2k(\log_k n)$	$2\log_k n$
Join	n	$k(\log_k n\text{-}1) - k + 2$	$\log_k(n\text{-}1) + 1$
Mass Join	$n + j - 1$	$k\log_k(n - j) + 2k\log_k j$	$\log_k j + 2\log_{2k} n$
Group Merge	n	$k(\log_k n - 1) + 1$	$\log_k n$
Key Refresh	$n - 1$	$k \log_k n - k$	$\log_k n - 1$
Exclusion	$2n$-2	$2k(\log_k n)$	$2\log_k n$

Table 6.2: Efficiency – computations

	Local exp.	Total exp.
IKA	$k(\log_k n) + 1$	$O(n\log_k n)$
Join	2	$n + 1$
Mass Join	$k(\log_k j) + 1$	$n - j + O(j\log_k j)$
Group Merge	2	$n + 2$
Key Refresh	2	$n + 1$
Exclusion	$k(\log_k n) + 1$	$O(n\log_k n)$

6.2.1 Join

Under the single-member join protocol, each node receives exactly one message. Thus the total number of messages is n.

It takes k simple rounds for a parent node to send messages to all of its children, and it takes 2 simple rounds for the root to receive a message from the joining member and send a message to the joining member. So the total simple rounds are $k(h - 1) + 2 = k(\log_k n\text{-}1) - k + 2$. However, it only takes one synchronous round for a parent to send messages to all of its children. So the total synchronous rounds are $(h - 1) + 2 = \log_k(n\text{-}1) + 1$.

Each node performs one exponentiation, except the joining member who performs two. So the total computation is $n + 1$.

6.2.2 Mass join

Under the tree-based mass join protocol, all existing members except the root receive exactly one message. Each joining member sends one message to its parent, and receives one message from its parent. So the total messages are $n - j - 1 + 2j = n - 1 + j$.

In phase 1 of the protocol, each joining member sends contributions to their parent node. This takes kh_j simple rounds and h_j synchronous rounds, where h_j represents the height of the joining members' tree. In phase 2, key pieces are propagated to all joining members. At the same time, key pieces are propagated to existing members. So this takes $kh_e + kh_j$ simple rounds and h synchronous rounds, where h_e is the height of the existing member's tree. Therefore, the total number of simple rounds is $2kh_j + kh_e = 2k\log_k j + k\log_k(n-j)$ and the total number of synchronous rounds is $h_j + h = \log_k j + \log_k n$.

Each existing member performs only one exponentiation. The number of exponentiations each joining member performs depends on the level, l, of the tree it is at (ie: the number of ancestors it has) and the number of right hand siblings it has, and the number of right hand siblings each of its ancestors have. Note that the leftmost node on the second to bottom level performs the most exponentiations. Since this stage of the protocol is almost exactly the same as the initial key agreement, the number of exponentiations each joining member performs is the same as the number of exponentiations performed during the initial key exchange. This number is:

$$k + 1 + l + \sum_{i=1}^{l}(k - x_i).$$

The worst case is $kh + 1 = k\log_k j + 1$. The total computations performed by existing members is $n - j$. The total computations performed by joining members is given by:

$$\sum_{all\ j}\left(k + 1 + l + \sum_{i=1}^{l}(k - x_i)\right) = O(j \log_k j).$$

6.2.3 Group merge

In the group merge protocol, each node receives one message. So the number of messages sent is n.

Assuming that only two groups are merging, let h_1 and h_2 be the heights of each group's trees respectively, and let $h_m = \max\{h_1, h_2\}$. Then, it takes one simple round for the roots of each tree to exchange messages, and $k(h_m - 1)$ additional simple rounds pass before the protocol

terminates. This is a maximum of $k(\log_k n - 1) + 1$ simple rounds. Similarly, it takes $\log_k n$ synchronous rounds.

Every node except the roots performs one local exponentiation. Each root performs two exponentiations. This is a total of $n + 2$ exponentiations.

6.2.4 Member exclusion

Since the member exclusion protocol involves performing an initial key agreement, the efficiency of the member exclusion protocol is identical to the initial key agreement protocol. The values are given in Table 6.1 and Table 6.2.

6.2.5 Key refresh

Under the key refresh protocol, every node except the root node receives one message, this is a total of $n - 1$ messages. There are $k(h - 1) = k \log_k n - k$ simple rounds and $h - 1 = \log_k n -$ 1synchronous rounds. Every node performs one exponentiation, excluding the root who performs two. The total exponentiations is $n + 1$.

7 Conclusion

The constraining environment of mobile military networks poses challenges to the design of an efficient and secure group key exchange protocol. While many solutions to handle group key exchanges exist, very few were found to be suited for such a network. Among distributory key exchange protocols, most required the use of a central trusted authority, which is unpractical in an ad hoc setting. We overviewed one distributory protocol, optimized rekey, which does not rely on a central authority, thus rendering it as a possible solution. However, it requires global broadcasts, an inefficient method of communicating when nodes may be several hops away from the broadcaster. In addition, the protocol suffers from the disadvantages of distributory protocols, including the need for secure communication channels and the absence of a fresh key guarantee.

Because of these disadvantages, we would prefer a contributory key agreement protocol. However, most existing contributory agreement protocols were found to be lacking. Some solutions made assumptions about the network topology, for example by requiring participants to form a ring as in Ingemarsson's protocol or a multi-dimensional cube in the Hypercube protocol. Other solutions, such as GDH.2, use a chained approach which is not as efficient as a tree-based approach. Yet even TGDH, while being an efficient tree-based protocol, still relied on global broadcasts.

We found that the best existing solution to key agreement in a tactical setting was the AT-GDH protocol. This protocol incorporated the efficiency of a tree structure without the need for global broadcasts and without making any assumptions about the network topology. The only problem was the AT-GDH protocol had not been fully specified. An algorithm existed for the initial key agreement, but no provisions were made for subsequent agreements which are needed when the group membership changes.

To complete the AT-GDH key agreement, we presented several auxiliary agreement protocols. All of the protocols stemming from additions to the group membership were found to be considerably more efficient that the initial agreement protocol. The key refresh

algorithm was also found to be similarly more efficient than the initial agreement protocol. However, we were not able to construct a satisfactory algorithm to handle exclusions that would provide an advantage over the initial agreement. This was due to the inherent structure of the key which impedes the efficient removal of any one participant's contribution. Furthermore, each protocol is secure in that they are resistant to passive eavesdropping, preserve backward and forward secrecy, and guarantee a fresh session key. If coupled with an authentication scheme, the protocols also become resistant to active attacks.

Possible future work may include modifying the initial agreement algorithm to produce a different key structure, thus allowing a more efficient member exclusion protocol. Another problem with the current protocol is no provisions are made to determine the shape of the spanning tree. For example, some nodes may have 100+ children, other nodes may have only two. Optimizing the shape of the tree would then optimize the protocol's efficiency. Since nodes on the left-hand side of the tree perform more computations than nodes on the right-hand side, even a perfectly balanced tree may not be computationally optimal.

A better solution to group key agreement in the tactical network may be found by exploring the use of a protocol other than Diffie-Hellman as the building block. Perhaps such a solution may be more efficient in general, handle member exclusions better, or provide a workaround to the bijection φ problem.

Bibliography

[AD02] E. R. Anton and O. C. M. B. Duarte. Group key establishment in wireless ad hoc networks. 2002. [On-line]. Available: http://www.gta.ufrj.br/ftp/gta/TechReports/AnDu02b.pdf

[ACTT01] D. A. Agarwal, O. Chevassut, M. R. Thompson, and G. Tsudik. An integrated solution for secure group communication in wide-area networks. In *Proc. Of 6th IEEE Symposium on Computers and Communications,* 2001.

[BD94] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology – EUROCRYPT '94,* volume 950 of *LNCS*, pages 275-286, Perugia, Italy, May 1994. Springer.

[BW98] K. Becker and U. Wille. Communication complexity of group key distribution. In *Proc. 5th ACM Conference on Computer and Communications Security,* pages 1-6, San Francisco, CA, November 1998. ACM Press.

[CBH03] S. Capkun, L. Buttyan, and J. Hubaux. Self-organized public-key management for mobile ad hoc networks. 2003. [On-line]. Available: http://icawww.epfl.ch/Publications/Capkun/CapkunBH03tmc.pdf

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory,* IT-22(6):644-654, November 1976.

[H01] M. Hietalahti. Key establishment in ad-hoc networks. 2001. [On-line]. Available: http://www.tml.hut.fi/~mhietala/KeyMan.ps

[Hie01] M. Hietalahti. *Efficient key agreement for ad-hoc networks.* Masters thesis, Helsinki University of Technology, Finland, 2001.

[ITW82] I. Ingemarsson, D. T. Tang, and C. K. Wong. A conference key distribution system. In *IEEE Transactions on Information Theory*, IT-28(5):714-720, September 1982.

[KPT00] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *ACM Conference on Computer and Communications Security,* pages 235-244, 2000.

[KPT02] Y. Kim, A. Perrig, and G. Tsudik. Tree-based Group Key Agreement. Cryptology ePrint Archive, Report 2002/009, 2002.

[LBPH99] M. Liu, J. Baras, S. Payne, and H. Harrelson. Modeling and simulation of large hybrid networks. In *Proc. 2nd Annual Advanced Telecommunications/ Infrastructure Distribution Research Program (ATIRP) Conference*, 1999.

[MAH00] S. Maki, T. Aura, and M. Hietalahti. Robust membership management for ad-hoc groups. In *Proc. 5th Nordic Workshop on Secure IT Systems (NORDSEC 2000),* Reykjavik, Iceland, October 2000.

[RBD00] O. Rodeh, K. P. Birman, and D. Dolev. Optimized group rekey for group communication systems. In *Proc. Network and Distributed System Security Symposium (NDSS'00),* San Diego, CA, Feb. 2000.

[SEM99] R. Sanchez, J. Evans, and G. Minden. Networking on the battlefield: Challenges in highly dynamic multi-hop wireless networks. In *Proc. IEEE MILCOM '99,* Atlantic City, New Jersey, October 1999.

[SSDW88] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A secure audio teleconferencing system. In *Advances in Cryptology – CRYPTO '88,* volume 403 of *LNCS,* pages 520-528, Santa Barbara, CA, USA, August 1988. Springer.

[Sti95] D. R. Stinson. *Cryptography Theory and Practice.* CRC Press, 1995.

[STW96] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to group communication. In *3rd ACM Conference on Computer and Communications Security,* pages 31-37, New Delhi, India, March 1996. ACM Press.

[STW98] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: a new approach to group key agreement. In *Proc. 18th International Conference on Distributed Computing Systems (ICDCS'98),* pages 380-387, Amsterdam, The Netherlands, May 1998. IEEE Computer Society Press.

[STW00] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. In *IEEE Transactions on Parallel and Distributed Systems,* 11(08), August 2000.

[WGL98] C.K. Wong, M. Gouda, and S.S. Lam. Secure group communication using key graphs. In *ACM SIGGCOM.* ACM, September 1998.

[ZH99] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. In *IEEE network,* November/December, 1999.

Acknowledgements

I would like to express my warm thanks to my major professor, Dr. Jim Davis, for without his guidance and encouragement the completion of this thesis would not have been possible. I would also like to thank Dr. Clifford Bergman and Dr. Doug Jacobson for their support as my committee members.

Finally, I would like to thank my parents and my husband Wayne for all their love and understanding.