I, Meetu Gupta, hereby submit this work as part of the requirements for the degree of:

Master of Science

in:

Computer Science

It is entitled:

“Minimization of Rekeying Overhead for a Secure and Scalable Multicast Framework”

This work and its defense approved by:

Chair: Dr. Dharma P. Agrawal
Dr. Ken Berman
Dr. James Caffery
Minimization of rekeying overhead for a secure and scalable multicast framework

A thesis submitted to the
Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in the Department of

Electrical and Computer Engineering and Computer Science
of the College of Engineering

August 2004

by

Meetu Gupta

B. Tech. (CS), Indraprastha University, Delhi, India, July 2002

Thesis Advisor and Committee Chair: Dr. Dharma P. Agrawal
Abstract

Multicast is an internetworking service that provides efficient delivery of data from a sender to multiple recipients. Security is an important issue in multicasting and is increasingly becoming a bottleneck. Security features like confidentiality, authenticity, and integrity can be provided with the help of cryptographic techniques like encryption. Access control to multicast services poses a huge problem and is an important issue to be addressed. A secret group key is used to prevent unauthorized access which has to be changed frequently in a dynamic membership environment. This results in a significant increase in the control overhead during the group key management process.

In this research, we reduce this overhead by using polynomial computations to distribute the group key. One of the existing frameworks, IOLUS, incurs a large overhead when a group member leaves the group. We propose a scalable approach in which this overhead is drastically reduced.
To My Dearest Parents and Sister
ACKNOWLEDGEMENTS

I wish to express my sincere thanks to my advisor, Dr. Dharma P. Agrawal, for the guidance that he provided throughout my work. He encouraged us to work in the areas in which we were interested. I would like to thank Dr. Berman and Dr. Caffery for being a part of my thesis committee. I am grateful to Dr. Anup Kumar for spending his invaluable time reviewing this work and for his suggestions to improve it.

I would like to thank all my lab mates, in particular, Anindo and Tarun for their help and encouragement during the course of my work.

My friends have made my life wonderful in Cincinnati. Special thanks to Gaurav, Mahesh, Rahul, Sachin, Dipti, Manjiri, Pranjal, and Akshay. I will cherish their friendship and all the good times we have shared.

I would like to express my gratitude to my parents, Anil and Vijay, and my sister, Ashu, for their constant support, love and encouragement.
Contents

List of Figures........................................................................................................................................ iv

1. Introduction........................................................................................................................................ 1
   1.1 Multicasting .............................................................................................................................. 2
   1.2 Applications of Multicasting................................................................................................. 4
   1.3 Applying security in multicast.............................................................................................. 5
   1.4 Purpose of the study............................................................................................................... 7
   1.5 Thesis Organization .............................................................................................................. 7

2. Security Concepts.............................................................................................................................. 8
   2.1 Basics of Network Security ................................................................................................. 8
      2.1.1 Security Services........................................................................................................... 9
      2.1.2 Security Attacks........................................................................................................... 10
      2.1.3 Security Mechanisms................................................................................................. 13
   2.2 Cryptographic Concepts......................................................................................................... 13
      2.2.1 Symmetric Encryption ............................................................................................... 13
      2.2.2 Public Key Encryption............................................................................................... 14
      2.2.3 Digital Signature and Certificate ............................................................................. 15

3. Secure Multicasting ........................................................................................................................ 17
   3.1 Secure Group ....................................................................................................................... 17
   3.2 Multicast Key Management.................................................................................................. 18
   3.3 Security Requirements ...................................................................................................... 19
   3.4 Efficiency Requirements..................................................................................................... 20
   3.5 Security and Efficiency Relationship................................................................................ 20
   3.6 Evaluation Criteria ............................................................................................................. 23
4. Related Work ........................................................................................................ 25
  4.1 Introduction ................................................................................................. 25
  4.2 IOLUS ......................................................................................................... 26
  4.2 Intra-domain Group Key Management .................................................... 29
  4.3 Kronos ....................................................................................................... 30

5. Level Based Key Establishment ...................................................................... 31
  5.1 Introduction ............................................................................................... 31
  5.2 Basic Ideas ................................................................................................. 32
    5.2.1 Mobility .............................................................................................. 32
    5.2.2 Authentication .................................................................................... 33
  5.3 Proposed Scheme ....................................................................................... 35
    5.3.1 Overview ............................................................................................ 35
    5.3.2 Join .................................................................................................... 36
    5.3.3 Transfer ............................................................................................. 38
    5.3.4 Leave ................................................................................................. 39
    5.3.5 Root Delegation Scheme (RDS) ........................................................ 41
  5.4 Performance Evaluation ............................................................................ 42
    5.4.1 Calculation of Average Level .............................................................. 42
    5.4.2 Simulation .......................................................................................... 45

6. Proposed Scheme ............................................................................................ 48
  6.1 Assumptions ............................................................................................. 48
  6.2 Overview ................................................................................................... 49
  6.3 Join Operation .......................................................................................... 50
    6.3.1 Rekeying after Join Operation ......................................................... 51
    6.3.2 Key Retrieval .................................................................................... 53
  6.4 Leave Operation ......................................................................................... 54
    6.4.1 Rekeying after Leave Operation ...................................................... 54
  6.5 Transfer Operation .................................................................................... 56
  6.6 Performance Analysis .............................................................................. 58
6.6.1 Security Analysis ................................................................. 58
6.6.2 Complexity Analysis........................................................... 59
6.6.3 Experimental Results ......................................................... 63

7. Conclusions and Future Work .................................................. 68
  7.1 Conclusions ................................................................. 68
  7.2 Future Work ................................................................. 69

Bibliography ................................................................................. 70
# List of Figures

1.1 Packet distribution using multicast and multiple unicast .................. 3

2.1 Types of Security Attacks ............................................................. 12

2.2 Symmetric Key Encryption ............................................................ 14

2.3 Public Key Encryption ................................................................. 15

3.1 A Secure Group ........................................................................... 17

3.2 Group Leader stores a key per member ........................................... 22

3.3 Group Member stores a key per member .......................................... 23

4.1 IOLUS multicast security framework .............................................. 27

4.2 Intra-domain Group Key Management Framework .......................... 29

5.1 Three way handshake .................................................................. 34

5.2 Organization of the level keys ....................................................... 35

5.3 Format of the Join Message .......................................................... 36

5.4 Analytical results for Probability vs. Level of a node ....................... 44

5.5 Effect on Average level as number of nodes increases ..................... 46

5.6 Effect of mobility on the number of re-key packets .......................... 47

6.1 Join Operation .............................................................................. 51

6.2 Leave Operation ........................................................................... 54

6.3 Transfer Operation ....................................................................... 56

6.4 Processing time for rekeying during a join operation ....................... 65

6.5 Processing time for rekeying during a leave operation ..................... 66
Chapter 1

Introduction

There are a number of applications which make use of multicasting because of its known advantages over unicast. Some of the applications are audio / videoconferencing, multiplayer online gaming, online/offline video distribution, news, and so on. There is a need to protect the packets being exchanged during the multicast. No member outside the group should have access to the information which is only meant for the group members. This is especially important for applications which are based on pay-per-view methodology.

Security is an important issue in multicasting and is increasingly becoming a bottleneck. If a multicast group is static, then nothing has to be done. However, in the case of a dynamic multicast group new research challenges have to be addressed. It should take into account the fact that a new member joining the group should not be able to decipher the previous conversation (forward confidentiality). Similarly, it should preserve the secrecy of the previous conversations and there should be no way for a member leaving the group to know the prior messages being exchanged within the group (backward confidentiality).
If the group members are mobile, the situation becomes even more complex. The members are not only allowed to join and leave the group but can also transfer from one area to another.

### 1.1 Multicasting

The concept of multicast was introduced by Steve Deering [1] in the late ‘80s. IP multicast is the transmission of IP datagrams to a host group. This host group consists of a set of zero or more hosts identified by a single IP destination address [2]. A host group is a collection of multicast capable hosts that can either transmit or receive datagrams. It is an efficient way to distribute data to a group of members and is particularly useful when multiple clients need the same kind of data at the same time where replicated unicast transmissions are extremely wasteful of network resources. Fig 1.1 illustrates this difference. If multiple unicasting is used, the server sends three identical data streams to the clients. On the other hand multicasting transmits a single copy of the message to the clients regardless of the number of clients. When a multicast enabled router receives this packet, it simply copies to the branches to which clients or multicast group members are located. This drastically reduces the bandwidth requirement for sending data to multiple clients.
The host groups are distinguished from one another by a specific Class D address assigned to them. The Class D addresses are reserved for multicast groups and are dynamically assigned to multicast groups. These addresses range from 224.0.0.0 to 239.255.255.255. The sender specifies the group class D address in the destination IP field as opposed to an individual IP address in unicasting.

Membership in a host group is dynamic and hosts are allowed to join or leave the network any time they want. IGMP (Internet Group Membership Protocol) is used for this purpose. It is used by the multicast routers to decide whether they should forward the data packet on a particular branch or not. IGMPv1 [2] is the one that is most
widely used and is specified in RFC 1112. IGMPv2 [4] and IGMPv3 [5] have also been proposed as enhancements to IGMPv1.

For delivering the data from the source to the destination nodes in other networks, a multicast tree is constructed. Some algorithms used for building such trees are simple flooding, spanning tree, reverse path broadcasting and reverse path multicasting. Some multicast routing protocols using these algorithms are Distance Vector Multicast Routing Protocol (DVMRP), Protocol Independent Multicast (PIM) and Multicast extension to Open Shortest Path First (MOSPF) [24-27].

1.2 Applications of Multicasting

Applications using group communications are gaining popularity at an increasing rate. Multicast is a more efficient method of providing group-oriented services as compared to unicast as it transmits the data to all the group members by transmitting a single copy. Some of the applications are video conferencing, stock monitoring, distance learning, military applications, warehouse monitoring, and news distribution. Some of the mobile commerce applications relying on multicast over wireless networks are mobile auctions, mobile entertainment services and, mobile advertising which uses asymmetric multicasting to send messages to certain users in certain locations. In a military environment, critical information is multicast to all the group members that may comprise of soldiers, war tanks, commandoes, and so on.
Multicast applications can be broadly classified into three categories: one-to-many (e.g., pay-per-view, stock price monitoring, and television broadcast), many-to-one (e.g., data collection, auctions, and polling) and many-to-many (e.g., distributed multiplayer games, teleconferencing, and chat groups) as specified in [6]. Multicast plays an important role in a wireless environment as the data is transmitted to all the members through a single transmission rather than multiple unicast transmissions. This leads to a drastic reduction in the energy consumed by the wireless nodes and also saves a lot of bandwidth. Thus, multicasting in a wireless environment is a very attractive option.

1.3 Applying security in multicast

Secure multicasting is needed for applications in which data is to be sent only to the members of the multicast group. Any member outside the group should not be able to decrypt the messages meant for the group. For example, suppose we stream a video which should be accessed only by the members who have paid for it. To prevent unauthorized access, a secret key is shared with all the group members. The sender then encrypts the video stream key using this secret and transmits it using multicasting. The group members can access the video by making use of this secret key given to them when they join the group. Since only the group members know this secret key, no outsiders can decrypt the video stream.

In a dynamic multicast group, the members can join or leave the group at any time. For such groups, the secret key has to be changed each time a new member wishes to
join the group to prevent it from gaining information about the previous conversation, also known as \textit{backward secrecy}. Similarly, when a member leaves the group, the group key has to be changed in such a way that the leaving member has no way of deciphering the new key and the future messages. This is known as \textit{forward secrecy}. This process is also known as ‘rekeying’ which involves creation and distribution of the new secret in a secure manner. Since the rekeying is done frequently i.e. on every join or a leave operation and affects a large number of hosts, it is important for it to be efficient. Some of the important parameters used to quantify the efficiency of a rekeying mechanism are processing time for key management, number of messages sent to inform all the members of the new key when a new member joins and an existing member leaves.

Hence applying security in multicasting poses additional challenges as compared to unicasting as the scheme used should be scalable, yet efficient. Threats include eavesdropping, the unauthorized creation of data, the unauthorized alteration of data, the unauthorized destruction of data, denial of service, and illegitimate use of data [7].

Several frameworks have been proposed in order to achieve secure multicast. Some of them are the IOLUS framework [8], the Nortel framework [9, 10] and the SRM Toolkit [11]. A survey of these frameworks is presented in [12].
1.4 Purpose of the study

In this thesis, we attempt to develop a secure, scalable and efficient scheme to minimize the control overhead involved in rekeying, thereby making it more practical.

Securing group communication in a way that the control overhead involved is independent of the group size is a challenging issue. The scheme should provide the desired level of security and should be scalable as the group membership is dynamic in nature. In IOLUS [8], a hierarchical scheme proposed for securing multicast in which the multicast group is divided into subgroups, either the processing time for key management or the number of messages sent for key update is linear in the size of the subgroup. Such schemes suffer from the scalability problem as the overhead involved in rekeying increases with the number of members in the multicast group.

1.5 Thesis Organization

We have divided this thesis into 7 chapters. Chapter 2 gives an insight into the security issues involved in communication networks and chapter 3 describes some of the issues involved in secure multicasting including the requirements and evaluation criteria. Chapter 4 discusses the details of the related work in secure multicasting. Chapter 5 discusses the details of our level based key establishment approach. We have presented our second approach and implementation details in chapter 6. We finally conclude in chapter 7 after presenting the future possibilities of our work.
Chapter 2

Security Concepts

This chapter provides the background information on security to understand the problems and the suggested solution in this thesis. We discuss the security issues in networks and multicasting that have to be taken into account for securing multicasting in an efficient manner.

2.1 Basics of Network Security

With the advent of computer networks there has arisen a need for network security which is primarily concerned with the protection of data being exchanged among a group of users. We look at the three aspects of network security [13]:

- **Security Service**: A service that ensures adequate security of the data processing systems and the information transfers. The services make use of security mechanisms to counteract an attack by an intruder.

- **Security Attack**: Any action that poses a threat to the security of information owned by an organization.

- **Security Mechanism**: A mechanism used to detect, prevent and respond to attacks by malicious users.
2.1.1 Security Services

The goals of securing networks can be represented by the following security services [13]:

- **Confidentiality:** It ensures that the information transmitted in the system is accessible only by the legitimate users and is never disclosed to unauthorized entities.

- **Authentication:** It allows a node to verify the identity of the nodes it is communicating with. If the authentication procedure fails, the malicious nodes can gain access to the information meant strictly for the authorized users.

- **Integrity:** It guarantees the validity of the messages being transmitted in the group. Thus, unauthorized users cannot modify or corrupt the data packets in any way. Modification includes writing, changing, changing status, deleting, creating and delaying or replaying of transmitted information.

- **Non-repudiation:** It ensures that neither the sending nor the receiving party can deny the transmission of a message.

- **Availability:** It requires that the intended network services be available to the authorized users when needed.
2.1.2 Security Attacks

The normal flow of information is depicted in the Fig 2.1(a). This flow can be disrupted by a variety of attacks by intruders who can either passively listen to information being transmitted or can alter the correct information. There are four general categories of security attacks [13]:

- **Interruption:** This attack is on the *availability* of network services. The intruder successfully prevents the receiver from accessing the information being transmitted in the system. This attack is pictorially depicted in Fig 2.1(b).

- **Interception:** This attack is on the *confidentiality* of the information. An unauthorized member gains access to a system resource meant only for legitimate users as shown in Fig 2.1(c).

- **Modification:** In this attack, the intruder compromises the *integrity* of the message by intercepting and sending a modified message to the receiver. As shown in Fig 2.1(d), the intruder, I, intercepts the original message $M$ and sends the modified message $M'$ to the receiver R.

- **Fabrication:** The attacker manages to send counterfeit information to the destination. The authenticity of the system is compromised as depicted in Fig 2.1(e) in which the intruder, I, sends a fabricated message $M'$ to the receiver.
The security attacks against a protected system can be classified into categories: passive and active attacks. In a passive active, the intruder mainly tries to gain access to the information being transmitted without altering it. Such an attack is harder to detect compared to an active attack. Using a passive attack the unauthorized user can get access to sensitive information. Even if the messages are encrypted, the intruder can analyze the traffic patterns to possibly determine the identity and location of the communicating hosts.

In an active attack the intruder may transmit messages, replay old messages, modify messages in transit, or delete selected messages. They can be classified into the following four categories:

1. **Masquerade**: The intruder pretends to be a valid user to gain access rights to the system and misuse the privileges given to the authorized users.

2. **Replay**: In this type of attack, the intruder records a conversation between a legitimate user and the host system to play it later to impersonate the user.

3. **Modification**: The attacker modifies the message being transmitted. Modification includes changing the contents of the message, dropping some of the data packets, reordering, and delaying of the packets.
4. **Denial of Service**: In this type of attack, the intruder disrupts the normal functioning of the system by flooding the server with messages or disabling the network.

![Diagram of Security Attacks]

- (a) Normal Data Transmission
- (b) Interruption of Data
- (c) Interception of Data
- (d) Modification of Data
- (e) Fabrication of Data

**Fig 2.1 Types of Security Attacks**

The objective is to detect, prevent and recover from the active attacks. Encryption is one of the ways in which such attacks can be prevented as only the legitimate users...
can access the information. Another security mechanism used is a digital signature which is used to provide proof of the sender.

2.1.3 Security Mechanisms

These mechanisms are used by the security services to provide a required security level to the system. There is no single mechanism that will provide all the services discussed above [13]. Cryptographic techniques like encryption and digital signatures are generally used to provide the means to counteract the attacks discussed. We focus on some of the cryptographic concepts in the next section.

2.2 Cryptographic Concepts

Message encryption [14] is used by the sender to convert plaintext into cipher text. It can provide a measure of authentication. It involves both an algorithm and a secret value known as the key. This can be done using either the symmetric key or public key encryption.

2.2.1 Symmetric Encryption

In symmetric or conventional encryption both the sender and the receiver share the same secret or key. The process is illustrated in Fig 2.2. Before the transmission of data begins, the sender and the receiver agree over a shared secret key $k$ for coding the messages. This key is exchanged over a secured communication channel. A plaintext message $m$ is encrypted by the sender using the secret shared key $k$ by applying an encryption algorithm on $m$. The resulting cipher text $c$ is then transmitted
to the receiver using a non–secure channel. On receiving the cipher text, the receiver decrypts the message using the key $k$.

![Symmetric Key Encryption Diagram](image.png)

**Fig 2.2 Symmetric Key Encryption**

### 2.2.2 Public Key Encryption

Public key encryption relies on the use of two different keys for encryption and decryption as compared to the single key used in the symmetric encryption. These two keys, also called the public/private key pair, are mathematically related. Even though the public key is related to the private key, this relation cannot be used to deduce the private key from a public key without the aid of additional information. The public key as the name suggests is publicly known to all the users whereas the private key is the secret key and is not shared with anyone. Encryption and decryption are two mathematical functions that are inverses of each other. The process of public key encryption is illustrated in Fig 2.3.
If a sender A wishes to send encrypted data to the receiver B, it first obtains B’s public key $PK_B$. This key is then authenticated so that A can verify that it actually belongs to B. The data is then encrypted using this authenticated public key of B. On receiving the encrypted message, B decrypts it using its private key $SK_B$. Since this private key is known only to B, no other user can decrypt the message sent by A. As compared to the symmetric key encryption, no secured channel is needed in this technique for distribution of keys. The disadvantage is that it needs more computational resources and is slower compared to symmetric key encryption.

**2.2.3 Digital Signature and Certificate**

In public key cryptography, the receiver B can verify the sender’s identity using *digital signatures*. The sender A attaches a signature with the message it sends. It generates this signature by computing the hash digest of the message and encrypting with its private key. To verify A’s signature, B computes the hash digest of the message and compares it with the message sent by A after decrypting it with A’s public key. If both match, B knows that the message has been sent by A.
Before sending the data, the sender has to obtain B’s authenticated public key. A has to assure that the key actually belongs to B and not to an intruder so that only B can decrypt the message. This authentication of the public key can be done with the help of digital certificates. A digital certificate is a statement issued by a third party named the certificate authority (CA) that verifies the public key of an entity. The recipient of an encrypted message uses the CA’s public key to decode the digital certificate attached to the message and obtains the authenticated public key of the entity.

These encryption techniques work well for unicast communications but multicasting is inherently more susceptible to various attacks [15]. In a multicast environment a large number of hosts participate which presents more opportunities for traffic interception. Furthermore, the multicast group addresses are publicly known, making the task of an attacker easier. We discuss some scalability problems and the security schemes proposed so far for multicasting in the next chapter.
Chapter 3

Secure Multicasting

In this chapter, we introduce the concept of a secure group and the basics of multicast key management. We also discuss the issues pertinent to key management for multicasting along with the evaluation criteria used for evaluating the key management solutions.

3.1 Secure Group

In a secure group, the group members know a secret value for encrypting and decrypting the data. Fig 3.1 illustrates the concept of a secure group.

Fig 3.1 A Secure Group
The data to be transmitted is encrypted by the sender using a group key which is known only to the group members. Since the knowledge of this key distinguishes a member from a nonmember, the management of this group key is an important issue. It involves distribution of the group key in an efficient and secure way to the group members. Another important issue involved in key management is rekeying. This ensures that the key is changed in an appropriate manner so that the past, present and the future communication is protected.

3.2 Multicast Key Management

The distribution of the keys is the main concern in a key management scheme. The solution should provide the desired level of security while keeping the communication and storage overhead low. Assuming that the appropriate encryption mechanism is used to transfer the data and the messages are not tampered with, the main problem is to distribute the session key in such a way that the unauthorized members cannot differentiate between a valid key and a random key. Thus, the problem of securing group communication reduces to the problem of securely distributing the key material and replacing older keys with new keys as the group membership changes. This is the multicast key management problem [16]. The key management problem can be broken down into the following two sub problems:

- **Key Agreement and Establishment**: The problem of mutually agreeing upon a secret value to be used for encrypting and decrypting the data.
• **Rekeying:** The problem of changing the group key according to the change in the group membership. When a new member joins or an existing member leaves, the key is changed to protect the information.

We concentrate on the rekeying problem in this thesis. We now look at the various security and efficiency parameters which should be considered for the design of an efficient key management solution.

### 3.3 Security Requirements

The main aim of secure multicasting is to ensure that only the authorized users have an access the packets sent to the group. In a dynamic group, the group membership changes frequently and rekeying has to be done frequently in a secure and an efficient manner. The security requirements for group management are:

- **Join / Backward Secrecy:** ensures a new member joining the group cannot gain access to the previous conversation. The key is changed after every join operation to provide this service in such a way that it is impossible to deduce the old keys using the current key.

- **Leave / Forward Secrecy:** ensures a leaving member does not have access to the messages being exchanged in the group. It also ensures that a member leaving the group cannot decipher the future key in any way except for brute-force attack.
• **Protection against Collusion:** ensures that nonmembers cannot collude to gain any information about the future messages or keys.

### 3.4 Efficiency Requirements

In a large group, efficiency becomes as important as security. The key management solution should consider the following efficiency requirements:

- **Traffic Volume:** The number of control packets should be minimized during the rekeying operation.

- **Processing Time for Key Management:** The computation time required for computing the new key and distributing it to all the members should be minimized.

- **Key Storage:** The number of keys stored at both the group controller and the group members should be minimized.

### 3.5 Security and Efficiency Relationship

There exists a tradeoff between security and efficiency [17]. It is important to consider the relationship between the security and efficiency requirements discussed above during the design phase of the scheme. As we will see with the help of a simple
example, a scheme designed to achieve a low communication cost might not be as efficient when evaluated for other requirements. This can be demonstrated with the help of the following two naïve key management schemes. This scheme is illustrated in Fig 3.2.

![Fig 3.2 Group Leader stores a key per member.](image)

In this naïve scheme, when a group member leaves the group, the group leader unicasts a packet containing the new key to the remaining members. In the example above, there are 100 nodes in the group and when node 1 leaves the group, the Group Leader (GL) sends out 99 rekey packets to each of the remaining members. Thus, a lot of communication cost is incurred in this case because the number of messages sent is dependent of the size of the group.

An alternative approach in which the communication is cut down to just one message during rekey involves giving out n-1 keys to every group member but not its own key.
These keys correspond to the rest of the members in the group. When the members are notified of a particular member leaving, they know which key is to be used for the future sessions. This scheme is illustrated in Fig 3.3.

In this complementary scheme all the group members store n-1 keys. When a group member leaves, the GL sends out a single packet which contains the new key encrypted with the key associated with the member who left. For example, when node 1 leaves the group, the GL encrypts the new key using $K_1$ which is known to all the members (2 to 100) except node 1. This saves a lot of communication overhead at the cost of making the scheme susceptible to collusion attack. Even if two members collude, they can decipher the future key. Key storage poses another problem increasing the storage complexity drastically. A member removed from the group can easily guess the new session key. Typically a large key size is chosen to increase the computational complexity of a ‘brute-force’ attack in which the user permutes

Fig 3.3 Group Member stores a key per member.
through all the possible combinations. In this scheme, the number of permutations reduces drastically and is equal to the number of members. This makes the task of finding the new key very easy in terms of computational complexity.

Thus, we see that there exists a trade-off between the security and efficiency requirements and they are highly inter-linked. A trade-off also exists between the communication cost and the key storage cost. An improvement made to the scheme from one perspective deteriorates the scheme when considered from another.

3.6 Evaluation Criteria

We will evaluate our scheme with the existing schemes based on the above observations on security and efficiency requirements as specified in [12]:

- **Scalability:** The scheme should be able to handle large dynamic groups with frequent key updates well. The control overhead should be independent of the number of group members.

- **Number of keys with a controller:** In a naïve scheme, the group controller shares a different key with each of its members apart from the common key used for data packet encryption. The controller stores \(n+1\) keys in such a case.

- **Number of keys with each group member:** The number of keys stored at each member. In a naïve scheme, like the complementary scheme discussed in
section 3.5, a group member stores a constant number of keys.

- **Join Secrecy:** A new member joining the group should not be able to decipher previous messages. In most of the schemes, it is easy to provide join secrecy in an efficient manner and is usually done by multicasting a single message encrypted using the current key to change the key.

- **Leave Secrecy:** When a member leaves the group, it should no longer have access to the messages being exchanged within the group. The key has to be changed following a leave operation. In a naïve solution, the update cost is linear in the size of the group.

- **Number of messages to update key on a join:** This parameter determines the number of update key or rekey messages sent when a new member joins the group. In most of the schemes, a single message is sent to the entire group informing them of the key to be used for future sessions.

- **Number of messages to update key on a leave:** In a naïve solution, the number of rekey messages sent when a member leaves the group is linear in the size of the group. A controller sends out $n$ messages to update the key.

- **Processing time for key management:** In a naïve scheme, the controller encrypts $n$ messages with the unique keys it shares with its group members.
Chapter 4

Related Work

In this chapter we provide a description of the related work in multicast security. A number of schemes have been proposed in the literature. Some schemes use a single common key for data packet encryption whereas others use different keys for different ‘subgroups’. We present a detailed description of the latter kind of schemes particularly IOLUS [8] on which our scheme is based.

4.1 Introduction

Several key management schemes have been presented to solve the problem of multicast security. These solutions can be categorized into three main groups [18]:

- **Centralized approach:** In this approach, there is a central entity called the Group Controller (GC). The GC manages the whole group and is responsible for distributing the key material to all the members during a rekey operation. Since a single GC performs the control operations, this scheme is not scalable for large groups.

- **Decentralized subgroup approach:** In this approach, the management of the group is simplified by dividing into subgroups. Each subgroup has a controller. Rekeying taking place in one subgroup has no effect on other
subgroups and thus localizes the effect making this approach scalable for large groups. It improves the fault tolerance of the system as there is no centralized group controller but the disadvantage is the reliance on trusted intermediary nodes.

- **Decentralized approach:** In this approach there is no explicit Group Controller. The process of key management is distributed among all the members of the group. This approach suffers from the problem of scalability as every member is participating in key generation.

We focus on the proposed decentralized subgroup approaches in this chapter as one of our proposed schemes follows this approach as well. We discuss three important schemes belonging to this subcategory.

### 4.2 IOLUS

IOLUS [8] uses a hierarchical approach where a group is divided into sub-groups that localizes the effect of rekeying. Each subgroup uses a unique subgroup key ($K_{SGRP}$) for encrypting the data which is known only to the subgroup members. This is achieved by having two types of special nodes which manage and connect the various sub groups. Each sub group is maintained independently by a sub group controller and is called the Group Security Intermediary (GSI). The various Group Security Intermediaries are controlled by a centralized Group Security Controller (GSC). The GSC coordinates the top-level GSIs and it is ultimately responsible for the security of
the overall group. The GSC and GSIs are together called Group Security Agents (GSA). Each GSI may have other GSIs below it to form a hierarchical network. The relationship between these agents is depicted in Fig 4.1.

When a host wants to send a message to the entire group, it unicasts the message to its GSI using the secret key it shares with it. The GSI decrypts the message, then encrypts it using the subgroup key $K_{SGRP}$ before multicasting it to its subgroup. Since only the subgroup members know this unique key, to send this message to other subgroups, the GSI re-encrypts it using the key it shares its parent and peer GSIs.

Now we give an overview of the algorithms used to add members, delete members and update the key during these processes.

Fig 4.1 IOLUS multicast security framework [8]
• **Join Operation:** When a host wishes to join the group, it sends a JOIN request to its designated GSI. If the GSI is not a member of the group, it first joins the group. The GSI upon receiving such a request, checks its ACL to see if the member is authorized to join the group or not. If the member is authorized, the GSI sends it a unique member key, $K_{MBR}$, which it shares only with this new member using a secure unicast channel. It also sends the new key generated using this member key.

*Rekeying:* The GSI generates a new subgroup key $K_{SGRP'}$. It encrypts this key using the old subgroup key $K_{SGRP}$ and multicasts it to its members.

• **Leave Operation:** A member may voluntarily leave the subgroup or may be expelled by the GSA. In both the cases rekeying has to be done.

*Rekeying:* The GSI generates a new subgroup key $K_{SGRP'}$, encrypts with the member key for each member and unicasts the message. This multiple encryption process is linear in the size of the subgroup. This provides forward secrecy as the leaving member does not know the member keys of other members in the subgroup.

This scheme has the advantage of localizing the effect of rekeying to a subgroup. It suffers from a few drawbacks. The control overhead involved in rekeying when a group member leaves is huge and is proportional to the number of members in the
subgroup. This is because the GSI encrypts the new key \( n \) times with the unique key it shares with every member. The rekey message cannot be encrypted with the old key as the leaving member knows the old key. A member addition procedure does not result in as much overhead since the new key is multicast to the remaining members by encrypting it with the old key which the incoming member does not know. A solution to reduce the overhead when a member leaves is proposed in the chapter 7.

### 4.2 Intra-domain Group Key Management

Hardjono et al proposed an intra-region group key management protocol (IGKMP) [10]. In IGKMP, the group is divided into administratively scoped *areas* [19]; there is one Domain Key Distributor (DKD) and several Area Key Distributors (AKDs). Each AKD is responsible for key distribution in an area. The DKD is responsible for generating the key for the entire group or domain and it propagates the group key with the help of AKDs. The DKD and all the AKDs in the group form an All-KD-Group as shown in Fig 4.2.

![Fig 4.2 Intra-domain Group Key Management Framework [18]]
This group is used to transmit the rekey messages. Unlike IOLUS, all the areas under a domain use the same key and hence the AKDs don’t have to re-encrypt the messages. This saves a lot of communication overhead however it makes the scheme susceptible to a single point of failure. If the DKD is compromised, the entire group is disrupted.

4.3 Kronos

In the approach proposed by Setia, Koussih and Jajodia [20] a membership change does not trigger the rekeying process. Instead, it is done periodically at fixed regular intervals. Kronos can be used within a distributed framework such as IGKMP. Although Kronos is similar to IGKMP, it operates in a different manner as the DKD is no longer responsible for generating the key. Each AKD generates the same group wise key and distributes it in its subgroup at the same time. The clocks of all the AKDs have to be synchronized to achieve this. All AKDs start with a master key $K$ and apply the same encryption algorithm to get the next key $K'$. This makes the network and processing overheads predictable.

There are several other proposed schemes which use a distributed subgroup approach for secure multicasting. A survey of such schemes is given in [18]. We discuss our proposed schemes in the next two chapters. We first discuss a completely distributed approach which uses the concept of level keys followed by our distributed subgroup approach.
Chapter 5

Level Based Key Establishment

We now present a security framework that is entirely distributed in nature [29].

5.1 Introduction

In our proposed scheme, a node is assigned a 'level' as soon as it joins the network. This procedure also establishes a level key between the node and its parent. The level key is the same as the one that the parent shares with its other children. The parent node passes on its level key to any node that wishes to join it. We thus have a tree in which each parent-children group share the same common level key. Note that this protocol is entirely different from the group based schemes that have area or subgroup keys [8]. In our scheme, an entire area need not be re-keyed whenever a "transfer" occurs, thereby drastically reducing the re-keying overhead.

The multicast group key update takes place by repeated decryption and encryption of the multicast key at each level. We thus achieve low communication overhead (since the update message would travel only once along each link), low latency (by the assumption that the routing layer has some cross layer information and can do this encryption-decryption process very fast without the packet going to the application layer), effective handling of user mobility (by introducing mobility parameters, as shown later), and most importantly, a security framework that is entirely distributed in nature.
We make the following assumptions for our work:

- All nodes that join a group can be either a sender or a recipient or both.
- A node that becomes a member has the computational power as well as the resources to generate a new communication key.
- Communication costs are much higher than the

We now introduce our basic ideas followed by the description of the proposed algorithm. We also present an analytical model of our scheme and also provide the simulation results.

### 5.2 BASIC IDEAS

#### 5.2.1 Mobility

One of the challenges facing the group key management schemes is modeling the concept of mobility. By incorporating mobility, we allow the members to not only leave and join the multicast group but also to “transfer” between locations. This transfer should be done in an optimal way to avoid re-keying the entire network and thus disrupting the communication and degrading the performance. We define a parameter called self mobility, $m_i$ for a node $i$ as:

$$m_i = \frac{\Delta n}{n_1 + n_2} \tag{1}$$
Where,

\( \Delta n \) = change in the neighborhood of node \( i \) in time period.

\( \Delta t = t_2 - t_1 \) i.e. the sum of number of nodes which have left and the number of nodes which have joined the neighborhood.

\( n_1 \) is the neighborhood at time \( t_1 \) and \( n_2 \) is the neighborhood at time \( t_2 \).

Thus, a large value of \( m \) indicates high mobility. This parameter is used during the join and the transfer operations. A node wishing to join a parent node \( k \) will calculate the mobility factor \( m_{ik} \) using \( m_i \) and \( m_k \) and the following relation:

\[
m_{ik} = m_i (1 - m_k) + m_k (1 - m_i) \tag{2}
\]

A low value of \( m_{ik} \) means that node 'i' would give a higher preference to node 'k'. The reason we choose such a relation is that we would like a low value (suitable) of \( m_{ik} \) only if either both the mobility values are high or both are low. This is because if one of the values is high and the other is low, a join would entail an immediate leave leading to a lot of re-keying overhead as a stable node is likely to have a lot of children. A closer inspection would reveal that in effect, equation (2) tries to 'pair' nodes with high mobility together so that the amount of re-keying is restricted to as few nodes as possible.

**5.2.2 Authentication**
The second idea deals with authenticating a node that is already in the group to another node in the same group to which it wishes to attach itself. This case might arise if a node (henceforth referred to as ‘A’) has moved to a different location and wishes to seek a transfer and attach itself to another parent (henceforth referred to as ‘B’). A similar situation might also arise if the parent has moved and the child wishes to reattach. The secret multicast key that they both know is K. The problem for A and B is to mutually authenticate each other by just knowing K and in a manner which would not allow a replay attack to take place.

In our scheme, A chooses a random number r and encrypts it using K and sends this secret to B. B decrypts r, chooses another number r’ and sends r’ and (r XOR r’) to A after encrypting it with K.

A decrypts (r XOR r’), finds r’ (by XORing with r), encrypts it using K and sends it back. Thus A and B authenticate each other. The advantage of this scheme is that r and r’ are unique random numbers which are different each time and hence there is no possibility of a replay attack. Also, a man-in-the-middle attack would not fetch the attacker anything as the secret K is never sent in this whole transaction. The only advantage that the attacker can achieve is to prevent A from getting attached to B. This would only add the overhead of a retrial to A's joining process. Fig 5.1 illustrates this mechanism.
5.3 PROPOSED SCHEME

5.3.1 Overview

With the above ideas in place, we now introduce our proposed scheme. The keys involved in our scheme are as follows:

1. \( K_m \): The common multicast key used to encrypt the data packets.
2. \( K_L \): The level key which is shared between a parent node and its children. For e.g., as shown in Fig 5.2, the level key \( K_{12} \) is shared between the parent node 1 and its children 4, 5 and 6. No other node in the group knows this key.
3. The public and private keys of a node.

Henceforth, we’ll refer to the lower numbered levels as the higher levels.

Each node of the multicast tree knows two level keys: one that it shares with its parent node and the other that it shares with its children as shown in Fig 5.2.
As soon as a new member joins the network or an old member leaves, $K_m$ is updated globally and $K_L$ is updated for the parent of the node that is leaving. In case an old member takes a transfer to a different region, $K_L$ is updated for the old and new parents of the node but $K_m$ remains the same. Now we describe the three basic operations: join, transfer and leave.

### 5.3.2 Join

When a new member (‘A’) wants to join the group, it sends out a join request (JOIN_REQ) message with the IN_GROUP bit (refer Fig 5.3) appropriately set to zero. The IN_GROUP bit indicates whether or not the node is already a member of the multicast group. This message is sent out in the normal multicast message format with the multicast group address set and a hop count specified. For our simulation, we chose the multicast hop count for the JOIN_REQ message to be 2 as this gives us a reasonable number of replies to choose from. The format of the join message is shown in Fig 5.3.
The field $E_{Kn}(r)$ is filled only when the IN_GROUP bit is set to 1. All the nodes receiving the JOIN_REQ message send a join reply (JOIN_REP) message with their mobility and level information. ‘A’ assigns a weight, $W_p$ to each member $p$ from which it receives a reply and sends a join confirm (JOIN_CONFIRM) message to the best node i.e. the one with the minimum weight. ‘A’ calculates the weight using the following relation:

$$W_p = aW_1 + bW_2 + cW_3$$  \hspace{1cm} (3)

Where,

- $W_1 =$ Degree of node $p$
- $W_2 =$ Hop count from ‘A’ to $p$
- $W_3 = m_{Ap}$, the mobility factor

The values of the coefficients $a$, $b$ and $c$ are highly application dependent. For instance, a network having high mobility should have a large $c$ while a sparse network should have a large value of $b$. A large value of $a$ makes the average degree of the network low, thereby forestalling multiple $K_L$ re-keying in the event of a join or a leave.

**Algorithm for Join Re-keying**
E_{a}(b) implies Encryption of ‘b’ using ‘a’ as the key.

Let ‘B’ = Future parent of ‘A’

Let K_{pa} = Public Key of ‘A’

1) ‘B’ generates new multicast key K_{m}'

2) ‘B’ multicasts E_{K_{m}'}(K_{m}')

3) ‘B’ generates K_{L'}

4) If (‘B’ is a leaf node) then
   ‘B’ sends E_{K_{pa}}(K_{L'}) to ‘A’

   Else

   ‘B’ sends E_{K_{L}}(K_{L'}) to all its children

A malicious node attempting to flood the system by sending numerous JOIN_REQ messages can be blocked by its neighbors. The neighbors keep a check on the number of request messages sent and drop the packets instead of forwarding them when this number reaches a threshold.

5.3.3 Transfer

As soon as the hop count of a node (‘A’) crosses a threshold, it tries to reattach to another parent (‘B’) which is at a higher level, using the JOIN_REQ message with the IN_GROUP bit set to 1. ‘B’ authenticates ‘A’ as described in section 5.2.2 with the JOIN_REQ, JOIN_REP and JOINCONFIRM messages containing the first, second and third messages respectively as shown in Fig 5.1. After sending the JOINCONFIRM message, the node sets its level to level(Parent) + 1 and sends a
CHANGE_LEVEL message to all its children. The main difference between the join and transfer operation is that the re-establishment of the multicast key, $K_m$ does not take place during the transfer operation. This reduces a lot of overhead and improves the performance as compared to Baseline Re-keying [28] in which the normal join and leave procedure is activated during transferring.

**Algorithm for Transfer Re-keying**

$E_a(b)$ implies Encryption of ‘b’ using ‘a’ as the key.

Let ‘B’ =Future parent of ‘A’

Let $K_{pa}$ = Public Key of ‘A’

Let $K_L$ ’= new Level Key

1) ‘B’ generates $K_L'$

2) If ‘B’ is a leaf node then:

   ‘B’ sends $E_{K_{pa}}(K_L')$ to ‘A’

Else

   ‘B’ sends $E_{KL}(K_L')$ to all its children.

   ‘B’ sends $E_{Kpa}(K_L')$ to ‘A’

**5.3.4 Leave**

The node wishing to leave the group first sends a rejoin request (REJOIN_REQ) to all its children. It then sends a leave request (LEAVE_REQ) message upstream to its parent node after receiving a REJOIN_CONFIRM message from all its children. The
leave request is signed by the node using its private key to prevent a malicious node from isolating a valid node from the group by impersonating it.

**Algorithm for Leave Re-keying:**

Ea(b) implies Encryption of ‘b’ using ‘a’ as the key.

Let ‘B’ = Future parent of ‘A’

Let KpC = Public Key of ‘C’

Let KL' = new Level Key

Let KL{Parent}i = Parent Level Key of i

Let KL{Child}i = Child Level Key of i

1) ‘B’ removes ‘A’ from its list of children.

2) ‘B’ generates KL'.

3) ‘B’ sends EKpC(KL') to every Child ‘C’.

4) ‘B’ generates Km' and sends EL'(Km'). downstream

5) ‘B’ sends EKL{Parent}B (Km'). upstream

6) For each node ‘C’ receiving a Km update message

   ‘C’ decrypts the new Km

If ‘C’ has a parent and the message was received from downstream, then

‘C’ sends EKL{Parent}C (Km') upstream

‘C’ sends EKL{Child}C (Km') downstream

The children of ‘A’ then use the join message with IN_GROUP bit set to 1 to attach to another member. Since the IN_GROUP bit is set to 1, only the level key is
changed. Also, a node that is already in the network attaches itself to a node that has a level higher than itself. This is to prevent cycles in the tree and also to maintain continuity of the tree.

We illustrate the leave operation by considering the case when node 1 wishes to leave in Fig 5.1. Node ‘1’ sends a REJOIN_REQ to nodes ‘4’, ‘5’ and ‘6’. These nodes then send a JOIN_REQ message with the IN_GROUP set to 1. The only possible nodes that can reply to these JOIN_REQ messages are nodes ‘2’, ‘3’ and ‘7’ (as they are at higher levels than ‘4’, ‘5’ or ‘6’). Upon reception of the JOIN_REP messages, ‘4’, ‘5’ and ‘6’ send the JOINCONFIRM messages to their new parent and then send a REJOINCONFIRM message to node ‘1’. Node ‘1’ now sends the LEAVE_REQ message upstream to node ‘2’. ‘2’ generates a new $K_m$ (i.e., $K_m'$) and a new level key (i.e., $K_{01}'$). ‘2’ sends $K_{01}'$ to ‘3’ and ‘7’ after encrypting it with their respective public keys. ‘2’ now encrypts $K_m'$ using $K_{01}'$ and sends it to ‘3’ and ‘7’. ‘3’ and ‘7’ then re-encrypt $K_m'$ using $K_{12}'$ and $K_{12}''$ respectively and send it downstream.

If the root node wishes to leave, it delegates the root status to one of its children and the change in the level is propagated down the tree.

5.3.5 Root Delegation Scheme (RDS)

Each time a node re-attaches itself to a level higher than itself, the nodes having higher levels suffer from a progressive increase in their degrees. To avoid this, when the root node’s degree crosses $D0_{AVG} + 1$ it promotes the level 1 node with the lowest
weight to level 0 by sending it a ROOT_DELEGATION message and setting its own level to 1. Here $D_{0_{AVG}}$ is the average level of the node at level 0. This level change is then propagated down the tree using the CHANGE_LEVEL message.

The simulation results show that the average degree of the network rapidly goes up by introducing this scheme.

### 5.4 PERFORMANCE EVALUATION

We first show that our scheme closely approximates the theoretical minima to the number of re-key messages.

In our scheme, for every leaving node, a re-key message is sent on a link exactly once. Thus we would obtain the theoretical minima if our multicast tree happens to be the minimum spanning tree.

As the network gets more mobile, the spanning tree gets distorted and we no longer have the optimal path. However, our scheme fares much better than almost all other schemes since for the same tree, our scheme would send the minimum number of messages as a message goes out on a link exactly once.

#### 5.4.1 Calculation of Average Level

In this subsection, we obtain a probabilistic estimate of the number of nodes at a particular level, thereby obtaining an estimate of the average degree. We compare
these values to those obtained through simulation. We also present an analysis of the effects of the Root Delegation Scheme on the average degree.

Assuming a node to have an equal probability of joining any other node in the group, the probability of the n\textsuperscript{th} incoming node to join the multicast group is given by 1/(n-1). It is evident that the first and second members of the group would have level 0 and 1 respectively. The third node can be at level 1 or level 2 with probabilities equal to 1/2. We define the function \(P_i(n)\) as the probability of the n\textsuperscript{th} incoming node to have a level i. Since a node can join the root with a probability of 1/(n-1), we get:

\[
P_i(n) = \frac{1}{n-1}
\]  

The probability of the n\textsuperscript{th} incoming node joining at level 2 is given by 1/(n-1) times the sum of the probabilities of the previous n-1 nodes joining at level 1.

\[
P_2(n) = \frac{1}{n-1} \sum_{i=1}^{n-1} P_i(i)
\]  

To obtain a general expression, the probability of the n\textsuperscript{th} incoming node to have level m is given by:

\[
P_m(n) = \frac{1}{n-1} \sum_{i=m-1}^{n-1} P_{m-1}(i)
\]
Where, $m \leq n$

The result (6) is very important as it provides a model for any multicast tree formation. Fig 5.4 shows the relationship between the probability and the level of a node. As can be seen, the probability peaks for levels of 1 and 2 and then rapidly falls down. The reason for this kind of a behavior is that a multicast tree formation is incremental in nature with nodes joining in one at a time.

Using (6), we can find an estimate of the average level for the $n^{th}$ incoming node.

$$L_{AVG}(n) = \sum_{i=1}^{n} i \times P_i(n)$$

**Fig. 5.4 Analytical results for Probability vs. Level of a node**

and,
\[ L_{AVG} = \sum_{j=1}^{n} \sum_{i=1}^{j} i \times P_i(n) \]  

(8)

Where,

\( L_{AVG} \) is the average level for the entire network.

The low value of \( L_{AVG} \) indicates the tendency of nodes to clutter towards the root node and hence justifies our Root Delegation Scheme (RDS). In the following section, it would be seen that simulation results readily concur with this observation. It would also be seen that the RDS increases the value of \( L_{AVG} \).

### 5.4.2 Simulation

We simulated the scheme in Network Simulator-2 (ns-2) with the number of nodes as 40 and the number of multicast nodes varying between 0 and 40. It is a discrete event simulator with support for various layers of the network protocol stack like Medium Access Control (MAC), Network, and Transport layers. The area of the considered scenario was 1000 m X 1000 m. Each simulation was executed for 600 seconds. Fig 5.5 shows the plots between the average level, \( L_{AVG} \) and the number of nodes in the group. The average level grows very slowly as the number of nodes increases.
As can be seen, in RDS, whose distribution closely resembles the binary tree level distribution, a node is neither confined to the higher layers, nor is its neighborhood too sparse. This means that during a level re-key, the number of messages and also the number of encryptions and decryptions would be low.

We also observed the effects of mobility on the number of level re-key and normal re-key messages. As can be seen from Fig 5.6, for low and medium mobility, the numbers of packets increase with an increase in the speed and a decrease in the pause time. However, at very high speeds, the routing layer is not able to deliver all the packets as the nodes are always on the move. Thus, most of the packets are lost, leading to retrials. The delays involved in this process become higher than the simulation time, leading to very few packets getting reported. We conclude this section by observing that the robustness of our scheme depends largely on the
robustness of the routing algorithm as shown by the simulation results. This would be true for any other protocol as well since any multicast protocol has to depend on the routing layer for timely packet delivery.

![Fig 5.6 Effect of mobility on the number of re-key packets](image)

We have suggested and analyzed a few basic ideas that can be employed while designing a security framework for a multicast ad hoc network. We’ve shown that in a mobile environment, mobility factors play an increasingly important role in establishing the structure of the network. We now look at our distributed subgroup approach which has a much better performance compared to IOLUS.
Chapter 6

Proposed Scheme

In this chapter, we present our distributed subgroup approach and provide a detailed description of the algorithm used to minimize the re-keying overhead. We have also evaluated the performance of our scheme based on the criteria outlined in chapter 3. The results obtained from the proposed approach are also compared with IOLUS. The analytical and experimental results show that our scheme has a better performance compared to IOLUS.

6.1 Assumptions

We have made the following assumptions in the design of our scheme:

- There are trusted nodes in the network called Group leaders (GLs) and are stationary.
- A GL has more computational power than a group member.
- A node that is already in the network can be trusted.

In the following sections we provide a detailed description of our scheme particularly the join, leave, transfer and rekeying operations.
6.2 Overview

We make use of polynomials to design and develop a secure, efficient and scalable rekeying scheme. We show using experimental results that polynomial computations are less expensive as compared to encryption and decryption operations. The aim of our scheme is to distribute the key related information in such a way that it is easy for the valid members to compute it and computationally infeasible to break the key for any entity outside the group.

The Group Leader (GL) chooses a random polynomial $p(x)$ of degree $t$ and distributes unique shares to all the members. This share distribution is done based on the techniques proposed in [21]. No group member knows the $p(x)$ polynomial. A new member is assigned a unique $ID$ value and is given the $p^{-1}(ID)$ value by the GL. The GL computes this share by substituting the member’s ID for $x$ in the $p^{-1}(x)$ polynomial i.e. $p^{-1}(ID)$. The ID and the privately shared values of a member are not known to any other entity. These secret values are used to retrieve the new group key after a rekeying operation. For example, if $p(x) = x^2 + 5x + 10$, then an incoming member is assigned a random ID (say $a_1$) and the private share $\frac{1}{(a_1^2 + 5a_1 + 10)}$.

After distributing the private shares of the polynomial, the GL then computes a valid member polynomial. This polynomial has the property that it evaluates to zero only for the valid group members and is computed in the following manner:

$$v(x) = (x-a_1) (x-a_2) (x-a_3) (x-a_t) \ldots (x-a_n)$$
Where,

\[ a_i \] is the ID of the valid group member \( i \).

By using this \( p(x) \) polynomial and a valid member polynomial \( v(x) \), the GL then constructs a rekey polynomial \( r(x) \) which is used to distribute the key every time a member wishes to join or leave the group. The polynomial \( r(x) \) is constructed in the following manner:

\[ r(x) = p(x) K + v(x) \]

Where,

\( K \) is the group key.

The GL broadcasts a re-keying message containing this \( r(x) \) polynomial. After a member receives the re-keying message, the members of the group can procure the key by evaluating the \( r(x) \) polynomial using their ID values and multiplying the result by their private share. The subsequent sections provide a detailed description of these operations.

### 6.3 Join Operation

The join process is illustrated in Fig 6.1. A node identifies the GL of the domain in which it is present through the beacon signal transmitted by the GL. Whenever a node wishes to join a multicast group, it sends out a JOIN_REQ message. The GL then checks its Access Control List (ACL) to verify if the member is authorized to join the
group or not. If the verification succeeds, the GL sends a JOIN_REP message which contains a randomly chosen ID & the private share of the node i.e. $p^J(ID)$. As discussed in section 6.2, these values are not publicly known. On receiving the join reply from a GL, the member sends a JOIN_CONFIRM message. After assigning a unique ID to its new host, the GL then prepares for the rekeying operation. It selects a new session key, denoted by $K'$ and distributes it to all its members using the rekeying polynomial.

![Join Operation Diagram](image)

1. JOIN Request
2. JOIN Reply
3. JOIN Confirm
4. Compute $r(x) = p(x) K' + v(x)$ and send

**Fig 6.1 Join Operation**

We now discuss in detail the polynomial operations involved in constructing the rekeying polynomial $r(x)$ at the GL and the operations performed by a group member to retrieve the key from $r(x)$.

### 6.3.1 Rekeying after Join Operation

The GL is responsible for coming up with a new random key $K'$ and distributing it to its members in a secure way. To do this, the GL first makes the required change in the
valid member polynomial \( v(x) \) so that it evaluates to zero for the new member as well.

For example, when a new member with ID \( c \) joins the group with two existing members with IDs \( a \) and \( b \) the \( v(x) \) polynomial is changed in the following manner:

The \( v(x) \) polynomial before the addition of the new member is:

\[
v(x) = (x-a)(x-b) = (x^2 - (a+b)x + ab)
\]

To account for the new member, the term \((x-c)\) is multiplied with the existing \( v(x) \) polynomial.

\[
v'(x) = (x^2 - (a+b)x + ab) \cdot (x-c)
\]

The GL then computes the rekeying polynomial, \( r(x) \) in the following manner:

\[
r(x) = p(x) \cdot K' + v'(x)
\]

Where,

- \( p(x) \) is the secret share polynomial
- \( K' \) is the new group key
- \( v(x) \) is the valid member polynomial

This polynomial is then broadcasted to all the group members. Note that no encryption operation is required during this process.
6.3.2 Key Retrieval

After receiving the rekeying message from the GL, the group member performs the following operations to retrieve the group key:

1. Evaluate the $r(x)$ polynomial at $x = ID$.

   $$r(ID) = p(ID) K' + v'(ID)$$

   As $v'(ID)$ evaluates to zero for a valid group member, the member is just left with the following expression:

   $$r(ID) = p(ID) K'$$

2. The next step is to multiply the above expression with their private share $p^{-1}(ID)$ to get $K'$. We make use of the mathematical property that the product of a polynomial with its inverse is unity.

   $$p(ID) p^{-1}(ID) = 1$$

   Thus,

   $$r(ID) p^{-1}(ID) = p(ID) K' p^{-1}(ID) = K'$$
The above polynomial operations performed by a group member are less expensive as compared to the decryption operation.

### 6.4 Leave Operation

The leave process is illustrated in Fig 6.2. Whenever a node wishes to leave the multicast group, it sends out a LEAVE_REQ message. After receiving such a request the GL performs the rekeying operation. It selects a new session key, denoted by $K'$ and distributes it to all its members using the rekeying polynomial. This procedure is similar to the rekeying process carried out during a join operation.

**Fig 6.2 Leave Operation**

We now discuss in detail the polynomial operations involved in constructing the rekeying polynomial $r(x)$ at the GL. The operations performed by a group member to retrieve the key from $r(x)$ are the same as during the join operation.

#### 6.4.1 Rekeying after Leave Operation
The GL is responsible for coming up with a new random key $K'$ and distributing it to its members in a secure way. To do this, the GL first makes the required change in the valid member polynomial $v(x)$ so that it does not evaluate to zero for the member leaving the group. For example, lets take a situation in which the nodes with IDs $a$, $b$ and $c$ are the members of a multicast group. When the member with ID $c$ leaves the group members the $v(x)$ polynomial is changed in the following way:

The $v(x)$ polynomial before the removal of the member is:

$$v(x) = (x-a)(x-b)(x-c)$$

To remove the new member, the term $(x-c)$ is dropped from the $v(x)$ polynomial.

$$v'(x) = (x-a)(x-b)$$

The GL then computes the rekeying polynomial, $r(x)$ in the following manner:

$$r(x) = p(x) K' + v'(x)$$

Where,

- $p(x)$ is the secret share polynomial
- $K'$ is the new group key
- $v'(x)$ is the valid member polynomial
This polynomial is then broadcasted to all the group members. Note that during a leave operation as well no encryption operation is required. This scheme performs much better than schemes which require $n$ encryption operations during a leave process to distribute the group key.

### 6.5 Transfer Operation

The transfer operation is illustrated in Fig 6.3.

![Fig. 6.3 Transfer Operation](image)

1. BEACON from different GL.
2. MOVED message to the new GL
3. Authenticate transferring node
4. Accept / Deny message
5. Compute $r(x) = p(x) K' + v(x)$ and send

A member detects its mobility using the periodic beacon signals sent out by all the Group Leaders. These beacon signals have the ID of the GLs. When a group member receives such a signal from a GL' different from its own, it knows it has moved into another region. It sends a MOVED message to its new GL' which contains the ID of the mobile host and the ID of its previous GL. The new GL then authenticates the
incoming node by communicating with its old GL. The rekeying is done in the mobile hosts’ previous subgroup but not immediately to avoid the ping-pong effect. The difference between the join and the transfer procedure is that during a transfer process, the area which the transferring node joins does not perform the rekey operation as the node was already a part of the group. The GL of the area which the node leaves performs a rekey operation to make sure that such a mobile node does not have more than one valid key at any given time.

The group members periodically send a KEEP_ALIVE message to the group leaders to renew their subscription. If a GL does not receive such a message on three occasions it removes the member’s name from its valid members list.
6.6 Performance Analysis

6.6.1 Security Analysis

In this section, we provide the security analysis of our scheme by evaluating its performance in the face of various attacks. The data packets are encrypted using the group key to make sure that the multicast session is robust against impersonation, interception, modification and fabrication attacks. We now consider the security requirements which are important for a multicast framework:

- **Protection against collusion** – Our scheme is a \( t \)-threshold scheme where \( t \) is the degree of the private share polynomial \( p(x) \). This means that unless \( t \) members collude no information regarding the key can be deciphered and the attacker will have to resort to ‘brute-force’ attack. \( t \) members can solve \( t \) equations to get information about the \( p(x) \) polynomial.

- **Backward Secrecy** – The group key is changed whenever a new member joins the group. This maintains the backward secrecy as the new group key is selected on a random basis and is not related in any way to the previous keys being used.

- **Forward Secrecy** – The group key is changed whenever an existing member leaves the group. This ensures that our approach provides forward secrecy as the group key chosen for a session is independent of the rest.
6.6.2 Complexity Analysis

We now analyze our scheme by finding the complexity of the algorithms being used for rekeying during the join and leave operations.

Setup:

During the setup process, the GL selects a random polynomial \( p(x) \) of degree \( t \), where \( t \) is the collusion threshold, and computes the polynomial \( p^{-1}(x) \). This inverse polynomial is used to calculate private shares to be assigned to the members. It then selects a unique group key \( K' \) and calculates \( p(x) K' \). Since \( t \) multiplications will be performed to compute this product, the complexity of this operation is \( O(t) \).

The GL then computes the valid member polynomial i.e. \( v(x) \). It selects \( t \) random values \( \{a_1, a_2, \ldots, a_t\} \) to compute \( v(x) \). \( v(x) \) is computed in the following way:

\[
v(x) = (x-a_1)(x-a_2)\ldots(x-a_t)
\]

The above computation involves \( \frac{t(t-1)}{2} \) multiplications and the same number of additions / subtractions. This is because multiplication with the \( i^{th} \) \( (x-a_i) \) term involves \( (i-1) \) multiplications and \( (i-1) \) additions. All the polynomial are stored as <degree, coefficient> values. Multiplying \( v(x) \) by \( x \) in the \( (x-a_i) \) term is equivalent to increasing the degree of all the terms by 1. For multiplying \( v(x) \) by \( a_i \) in \( (x-a_i) \) term, \( (i-1) \) multiplications have to be performed with \( a_i \) and \( (i-1) \) additions/subtractions have to
be performed. Thus, for $t$ such terms, the total number of multiplications performed is:

$$\sum_{i=1}^{t} (i - i) = 1 + 2 + 3 + \ldots + (t - 1) = \frac{t(t-1)}{2}$$

Thus, the complexity to compute $v(x)$ during the initial setup phase is $O(t^2)$. The selection of $t$ IDs is done to make sure that the degree of $v(x)$ is $t$ so that the coefficients of the higher degree terms in $p(x)$ are not revealed. The first $t$ incoming members are assigned an ID from these $t$ IDs. Since $v(x)$ already has the corresponding $(x-ID)$ term, the processing overhead is less during the first $t$ joins.

Next, the rekeying polynomial, $r(x)$ is calculated using the following equation:

$$r(x) = p(x) K' + v(x)$$

This operation involves $t$ additions as there are $t$ terms in both $p(x)$ and $v(x)$ and thus the complexity is $O(t)$.

### Join Operation:

When a new member joins the group, the GL assigns it a unique ID and $p^{-1}(ID)$. This polynomial evaluation can be done in $O(t)$ steps using Horner’s Rule. The GL then computes the rekeying polynomial $r(x)$ to distribute the new key. It selects a new random group key $K'$ and calculates the product $p(x) K'$. During the first $t$ joins, the $v(x)$ polynomial does not change. Hence, $O(t)$ multiplications and $O(t)$ additions have to be performed to compute the new $r(x)$. 

When the \((t+1)th\) member joins the group, \(v(x)\) is changed and \((x-a_{t+1})\) is added:

\[
v'(x) = v(x) (x-a_{t+1})
\]

The polynomial is stored as \(<\text{degree, coefficient}>\) values. Multiplying \(v(x)\) by \(x\) in the \((x-a_{t+1})\) term is equivalent to increasing the degree of all the terms in \(v(x)\) by 1. For multiplying \(v(x)\) by \(a_{t+1}\) in \((x-a_{t+1})\) term, \(t-1\) multiplications have to be performed with \(a_{t+1}\). \(v'(x)\) can then be computed after \(O(n)\) additions/subtractions. Thus, \(O(n)\) multiplications and \(O(n)\) additions/subtractions are performed where \(n\) is the number of remaining group members.

The rekeying polynomial, \(r(x)\), is then computed by adding \(p(x) K'\) and \(v'(x)\). In this last step \(O(n)\) additions are done to compute \(r(x)\). Thus, a total of \(O(t+n)\) multiplications and \(O(2n+t)\) i.e. \(O(n)\) additions are performed where \(n\) is greater than \(t\).

**Leave Operation:**

When a member with ID \(a_L\) leaves the group, the GL has to compute the rekeying polynomial \(r(x)\). It selects a new random group key \(K'\) and computes \(p(x) K'\). If the number of remaining members in the group is not less than \(t\), the GL removes the \((x-a_L)\) term from the \(v(x)\) polynomial. This removal is done by dividing by \((x - a_L)\) in the following way. Let us say,
\[ v(x) = a_t x^t + a_{t-1} x^{t-1} + a_{t-2} x^{t-2} + \ldots + a_0 \]

Now, we can say that

\[ v(x) = (x - a_L) v'(x) \]

\[ a_t x^t + a_{t-1} x^{t-1} + a_{t-2} x^{t-2} + \ldots + a_0 = (x - a_L) (a'_{t-1} x^{t-1} + a'_{t-2} x^{t-2} + \ldots + a'_0) \]

Multiplying by \( x \) in the \((x - a_L)\) term is equivalent to increasing the degree of all the terms in \( v'(x) \) by 1. Thus, a total of \( t \) multiplications are performed with \( a_L \). The coefficients of the terms in \( v'(x) \) can then be computed by using the following equations:

\[ a'_{t-1} = a_t \]
\[ a'_{t-2} = a_{t-1} + a_L a'_{t-1} \]
\[ a'_{t-3} = a_{t-2} + a_L a'_{t-2} \]

Thus, the general formula for the \( i^{th} \) term is:

\[ a'_{t-i} = a_{t-(i-1)} + a_L a'_{t-(i-1)} \]

Thus, \( v'(x) \) can be computed using \( O(t) \) subtraction/comparison operations. Hence, a total of \( 2t \) or \( O(t) \) multiplications and \( O(n + t) \) additions/subtractions have to be performed to compute the new \( r(x) \).
If the number of members left in the group is less than \( t \), an extra operation is performed. The GL chooses a random value \( a_t \) and multiplies \( v(x) \) with \((x - a_t)\). This is done to ensure that the degree of the \( v(x) \) polynomial is greater than or equal to the degree of the \( p(x) \) polynomial so that its higher degree coefficients are not revealed. Thus, in such a case, the computation of \( v(x) \) involves \( 2t \) operations and hence the complexity is \( O(t) \).

The rekeying polynomial, \( r(x) \), is then computed by adding \( p(x) K' \) and \( v'(x) \). In this last step \( O(n) \) additions are done to compute \( r(x) \). A total of \((n + t)\) multiplications and \((2n + t)\) additions are performed. Thus, the complexity is \( O(n) \).

### 6.6.3 Experimental Results

We have performed simulations to evaluate our scheme using the evaluation criteria discussed in section 3.6. The algorithms have been implemented in C.

- **Number of keys with a controller**: The controller stores the group session key and the ID it assigns to the group members. In IOLUS, the controller stores \((n + 1)\) keys.

- **Number of keys with each group member**: A group member has to store one extra value compared to IOLUS. It stores three values: the group key, its own ID and the private share \( p^{-1}(ID) \).
• **Number of messages to update key on a join:** A single message is sent to the entire group to inform them of the new group key to be used for future sessions. In IOLUS, two messages are sent, one using the old multicast key and the other using the incoming member’s key.

• **Number of messages to update key on a leave:** A single message is sent to the entire group which contains the rekeying polynomial whenever a group member leaves. In IOLUS, either n messages are unicast to the members or one large message is sent which contains the group key encrypted with the individual keys of all the members.

• **Processing time for key management:** A comparison of the processing overhead with IOLUS when a member joins the group is shown in Fig 6.4. The overhead is less in the case of IOLUS as it has to perform only two encryptions to distribute the group key to all the members. One encryption is done using the previous key to distribute the new key to all the existing members and the other encryption is done to give the new key to the incoming member. In our scheme, this overhead varies with the threshold $t$ we choose. In the graph, the value of $t$ is chosen to 10, 20 and 30 i.e. we take the case when 30% of the nodes in the group are malicious.
Fig 6.4 Processing time for rekeying during a join operation

In IOLUS, the controller encrypts $n$ messages with the unique keys it shares with its group members when a group member leaves the group. In our scheme, we use polynomial operations to replace this encryption process to distribute the group key. The operations performed by the group controller and the group members are polynomial multiplication, addition and subtraction. Our experimental results show that by using polynomial operations for distributing the key we outperform other schemes which encrypt $n$ times. This is shown in Fig 6.4. In our scheme, the processing time increases as the threshold $t$ is increased. In this graph, the value of $t$ is chosen to be 10, 20 and 30. A value of $t$ equal to 10 implies that 10 members have to be malicious and collude to launch an attack. It can be seen from the graph that even if a provision is provided to protect the group communication against 30% malicious
nodes, the processing time at the GL is still much lower compared to Advanced Encryption Standard (AES). Our scheme is compared against the AES encryption algorithm whose performance is best compared to other encryption algorithms like DES, IDEA, CAST, RC4 [22]. The keys for these algorithms are of size 128 bits and an optimal C implementation of AES is used [23].

![Processing overhead during Leave Operation](image)

**Fig 6.5** *Processing time for rekeying during a leave operation*

- **Total processing overhead during rekeying:** It can be observed from both the graphs, that after a series of joins and leaves in a multicast group, the processing time is much more compared to our proposed approach.
• **Scalability:** It can be observed from Fig 6.5 that on increasing the number of nodes per GSI, the processing time for IOLUS increases linearly. In our scheme, this overhead does not increase in such a sharp manner making it more scalable.
Chapter 7

Conclusions and Future Work

In this chapter, we conclude our thesis work and suggest some possible extensions for future work. There are some aspects that we have not addressed and could possibly be researched in future.

7.1 Conclusions

In this thesis work, we have addressed the multicast security problem. Security includes properties such as confidentiality, integrity, authenticity. In addition to these requirements additional requirements are introduced in group or multicast communications. These are forward message secrecy, backward message secrecy and prevention from collusion. A lot of work has been done in this field but most of the schemes rely on the standard encryption and decryption processes to provide the required level of security. Whereas encryption is needed for data messages in the network, we show in our work that if polynomial computation is used for control packets, the control overhead can be drastically reduced. We minimize the rekeying overhead in our scheme while restricting the number of rekey messages to one during both the join and the leave operation. The simulation results show that the processing time is much less as compared to the encryption and decryption operations as the number of members in the multicast group increase and thus, makes it more scalable as compared to IOLUS.
7.2 Future Work

A lot of solutions have been proposed for securing group communications. There is no one solution which is suitable for all the applications. Designing a scheme which balances all the security and efficiency requirements is an open issue because of the inter-dependencies.

One of the challenges facing the group key management schemes is modeling the concept of mobility. Smarter key management schemes can be developed by developing and utilizing a probabilistic model for group member mobility and by clustering the group members according to their mobility. If members with low mobility are in the same group, stable zones in the group are created. This would restrict the rekeying to as few subgroups as possible but will increase the computations to be performed by the GL of a high mobility subgroup. It would be interesting to apply these concepts to the distributed subgroup approach.
Bibliography


