AN EFFICIENT AUTHENTICATED GROUP KEY AGREEMENT ON TREE-BASED BRAID GROUPS IN MOBILE AD HOC NETWORKS

Thanongsak Aneksrup

A Dissertation Submitted in Partial
Fulfillment of the Requirements for the Degree of
Doctor of Philosophy (Computer Science)
School of Applied Statistics
National Institute of Development Administration
2010

AN EFFICIENT AUTHENTICATED GROUP KEY AGREEMENT ON TREE-BASED BRAID GROUPS IN MOBILE AD HOC NETWORKS

Thanongsak Aneksrup

School of Applied Statistics

Associate Professor.......[illegible]....... Advisor

(Pipat Hiranvanichakorn, D.E.)

The Examining Committee Approved This Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science).

Assistant Professor.......[illegible].......Committee Chairperson

(Pramote Kuacharoen,Ph.D.)

Associate Professor.......[illegible].......Committee

(Pipat Hiranvanichakorn, D.E.)

Assistant Professor.......[illegible].......Committee

(Ohm Sornil, Ph.D.)

Assistant Professor.......[illegible].......Committee

(Tossapon Boongoen, Ph.D.)

.......L. Bosuwan.......Dean

(Lersan Bosuwan, Ph.D)

April 2011

ABSTRACT

Title of Dissertation	An Efficient Authenticated Group Key Agreement on Braid Groups in Ad Hoc Networks
Author	Thanongsak Aneksrup
Degree	Doctor of Philosophy (Computer Science)
Year	2010

As various applications of ad hoc networks have been proposed, security issues have become a central concern and are increasingly important. This research proposed a contributory group key management that approaches by using braid groups, key tree and the identity authentication system. Without any assumption of prefixed online trust relationship between nodes, the proposed method works in a self-organizing way to provide the key generation and key management services using Tree-based Braid Groups. The using of the proposed Tree-based Braid Groups has following advantages: (1) the communication cost is minimized to constant time; (2) the complexity of computation is decreased to linear since the braid groups using just product and inverse operation by avoiding modular exponential operation. The using of identity authentication has the following advantages: (1) the storage space and the communication overheads can be reduced in that the certificate is unnecessary; (2) the computational costs can be decreased since it requires no public key verification; (3) there is no key escrow problem since the certificate authority (CA) does not know the user's private keys. The proposed protocols are more simple, secure and efficiency for group key management in mobile ad hoc networks.

ACKNOWLEDGEMENTS

I would like to express sincere thanks to my advisor, Associate Professor Dr. Pipat Hiranvanichakorn, for his valuable advice, encouragement and guidance in making this dissertation. I also wish to extend thanks and appreciation to all of the committee members, Assistant Professor Dr. Pramote Kuacharoen, Assistant Professor Dr. Ohm Sorni and Dr. Tossapon Boongoen for their comments and suggestions. Thank is also dedicated Dr. Supachot Lerdwarathum as my close friend and classmate. Family, special thanks are due to my wife for supporting me and taking care of our children, extremely well throughout my study.

Thanongsak Aneksrup

April 2011

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Security on Mobile Ad-hoc Networks

Mobile ad-hoc networks (MANETs) are special type of wireless network in which a collection of mobile devices with wireless network interfaces may form temporary network, without the aid of any fixed infrastructure or centralized administration. A whole new generation of portable devices is commercially available, such as personal digital assistants (PDA), computer laptops, etc. In MANETs, nodes within their wireless transmitter range can communicate with each other directly, while nodes outside the range have to rely on some other nodes to relay messages. Thus a multi-hop scenario occurs, where packets of the source node are relayed by several intermediate nodes before reaching the destination node. Every node what the packet passed functions to be as a router. The success of communication highly depends on the other node’s cooperation. The MANETs are also regarded as ideal technology for creating instant communication networks for civilian and military applications. In recent years, MANETs have received a great deal of attention in both academia and industry. This emerging technology aims to provide “anytime-anywhere” networking services on a potentially large scale.

While MANETs can be quickly and inexpensive setup as needed, security is a critical issue compared to wired or other wireless counterparts. Many passive and actives security attacks could be launched from the outside by malicious nodes or from the inside by compromised nodes. There are five fundamental security issues which have to be addressed: confidentiality, integrity, non-repudiation, authentication and availability. Because of the high level of self-organization, dynamic topology, dynamic membership or vulnerable wireless link, MANETs are difficult to secure. In addition, security solutions applied in most traditional network with a static configuration may not directly implement for protecting them according to the mobility of MANETs.

Many applications of MANETs involve collaborative computing among a large number of nodes and are thus group-oriented in nature. Examples of such applications include coordination of fire fighters in a rescue task and coordination of soldier during battle. For deploying such applications in an adversarial environment such as battlefield or even in many civilian commercial scenarios, it is necessary to provide support for secure group communication.

Secure group communication requires scalable and efficient group membership management with appropriate access control measures to protect data and to cope with potential compromises. To this end, a secret key for data encryption must be distributed securely and efficiently to current members. Each time a membership change occurs, the secret key must be changed to ensure backward and forward secrecy. Several proposals for group key management have been made recently. They range from key distribution schemes for large-scale single-sender multicast to contributory key agreement schemes for small any-to-any peer groups. Although most of them focus on wired networks, extensions to wireless networks (and MANETs) should be explored as such networks are becoming more common place. Due to the lack of fixed infrastructure and limited resources, it will be much more complex to adapt protocols and other technologies from the infrastructure based networks.

The importance of secure group communication motivates the needs of common shared group key. There are three group key management schemes including centralized, distributed and contributory group key management. First scheme, the group key establishment can be centralized, where an entity is responsible for generating the group key and distributing to group members. This approach has advantage of being simple but it is claimed that is not appropriate for dynamic group communication since the central key server must be, at the same time, continuously available and present in every possible subset of a group in order to support continued operation in the event of arbitrary network partitions. Continuous availability can be addressed with fault-tolerance and replication techniques. Unfortunately, the omni-presence issue is impossible to solve in a scalable and efficient manner. Second scheme, distributed group key management is more suitable to group communication, especially over unreliable networks. It involves dynamically selecting a group member that act as a key server. Although robust, this approach has a notable drawback in that it requires the key server to maintain long-term pairwise secure channels with all current group members in order to distribute group keys. Third scheme, contributory group key agreement requires each group members to contribute for the group key creation. This approach is fault tolerant and diminishes the risks of potential vicious key generating by a single entity to avoid the problem with the single point of trust and failure.

In MANETs, key establishment protocols should also provide forward and backward secrecy, because the dynamic nature allows the joining and the leaving of member nodes. Most key establishment protocols are based on Diffie-Hellman key exchange protocol. Many group key agreement protocols were designed based on the natural extension of Diffie-Hellman key exchange protocol to the multiparty case, while inheriting all its security characteristics and providing a contributory group key establishment.

The purpose of this research is to find an authenticated, secure and efficient key agreement protocol for a group communication in MANETs. The protocol was based on braid groups cryptographic and tree key techniques by using fully distributed authenticated without trusted third party such as CA. The protocol requires an off-line key server for approving public key as CA. The required computational processes in braid groups techniques are much faster than Diffie-Hellman and elliptic curve cryptographic techniques. The braid groups is applied in protocol just product and inverse operation by avoiding modular exponential operation. Therefore, the protocol reduces both communication and computation cost.

1.2 Related Works

Recently a number of protocols have been proposed to solve the problem of key management over wireless ad hoc networks. Key pre-distribution has been discussed in several approaches. Zhu, Xu, Setia, and Jajodia (2003) discuss a secure communication between two nodes in an ad hoc networks using probabilistic key sharing scheme in which enables two nodes to establish a pairwise shared key. The protocol used the off-line key server in key pre-distribution phase what is used to initialize all the nodes. Their protocol is based on two techniques including probabilistic key sharing and threshold secret sharing. A password based multi-party key agreement scheme was proposed Asokan and Ginzboorg (1999) where all the nodes are assumed to share a password. Basagni, Herrin, Rosti, and Bruschi (2001) described a secure ad hoc network in which all the nodes share a group identification key stored in tamper-resistant devices. Though all the above schemes perform efficiently, they require that all the nodes have some pre-determined knowledge. In ad hoc networks where mobile nodes do not grant the privilege of knowing other group members beforehand, assumption of such a pre-shared secret is invalid.

The concept of mobile certificate authorities has been discussed by Yi and Kraverts (2002) and Kong, Zerfos, Luo, Lu, and Zhang (2001). In such schemes, the responsibilities of a CA are distributed among a set of wireless nodes. A subset (threshold) of such CAs must be contacted to obtain a valid certificate. Such schemes have several advantages such as providing data integrity, authentication and non-repudiation. The drawbacks of such schemes are: (a) identifying nodes that perform the role of the CAs, consequently these nodes must spend more power; (b) constant availability of a threshold of CAs in a mobile network; and (c) the use of the computationally expensive public key encryption systems. Public key certificates are also used by Hubaux, Buttyan, and Capkun (2001), where all the nodes are assumed to maintain a local certificate repository and a probabilistic method is used to achieve a certificate chain between two nodes. This scheme requires that all nodes are preloaded with a set of certificates and it is possible that two nodes in the ad hoc network do not achieve a certificate chain. Also, the authors in (Yi et al., 2002; Kong et al., 2001; Habaux et al., 2001) did not address the distinct features of secure group communication such as group key formation and member join or leave.

Key establishment using contributory key agreement protocols are discussed following. Anton and Duarte (2002) discussed a number of such protocols previously used on wired networks and concluded that the CLIQUES protocol suite (Steiner, Tsudik, and Waidner, 1998) was best suited for ad hoc networks. Li, Wang, and Frieder (2002) also used the GDH (Group Diffie-Hellman) protocol, part of the CLIQUES protocol suite, for key agreement over ad hoc networks. GDH is an efficient protocol with good support for member join and leave operations but it has some unfavorable features with regard to ad hoc networks. Most importantly, the GDH scheme requires that the members be serialized or structured in order to compute the group. Also, the last member in the group acts as a Group Controller (GC). Consequently the GC does more computation than the other members in the group. Thus, in using GDH for ad hoc networks deciding which member is going to perform the operation of a GC is an important problem. Kim, Perrig and Tsudik (2000; 2004) adapted it to a contributory key agreement protocol TGDH. Every group member creates a key tree separately. Each leaf node in key tree was associated with a real group member, while each non-leaf node corresponds to a subgroup of group G, considered a virtual member. The protocol constructed the keys in key tree, which every node on the key tree has a Diffie-Hellman key pair. The number of exponential in computation overhead is $\log_2(N)$The important problem of their approach is that the number of message in partition event is in the order of $\log_2(N)$. Another group key agreement developed for teleconferencing was proposed by Steer, Strawczynski, Diffie and Wiener (1988) in naming STR protocol. This protocol was of particular interest since the structure of its group key form a special case of the TGDH. STR is efficient for joining new group members as it takes only two rounds and two modular exponentiations. Member leaving, however, is relatively difficult. Due to the small number of rounds which results in a low communication overhead. Kim, Perrig, and Tsudik (2001) extended the STR protocol. Their researches have constructed protocol that supports dynamic group. However, its computation costs were quite expensive because the exponential depended on amount of member, but communication costs were constant round on all membership events and were not depending on the amount of members. The main disadvantage of TGDH and STR protocol is single point of failure at sponsor what had to existing in key tree. These protocols do not support this situation.

Steiner, Tsudik, and Waidner (2000) proposed a family of Group Diffie-Hellman (GDH) protocols for dynamic peer groups. Based on them, Ateniese, Steiner and Tsudik (2000) proposed a new multiparty authenticated key agreement protocol, which offers key authentication or integrity, key confirmation, and non-repudiation of group membership. However, some flaws in this protocol have been found by Pereira and Quisquater (2001). Lee, Lui and Yau (2006) extended tree-based group Diffie-Hellman on distributed key agreement protocol, reducing the rekeying complexity by performing interval based rekeying including rebuild algorithm, batch algorithm and queue-batch algorithm. They also presented an authenticated key agreement protocol. As the success of their scheme was partially based on a certificate authority, their protocol encountered the same problems as centralized trust mechanisms. Yasinsac, Thakur, Carter, and Cubukcu (2002) proposed a key agreement protocol using the Diffie-Hellman key exchange concept. The main advantage of this protocol was that it did not involve member serialization. On the downside, the protocol did not efficiently support member join/leave operations and the protocol also involved the services of a group controller.

In Wang and Wu (2006) and Abdel-Hafex, Miri, and Orozco-Barbosa (2004) the protocols were based on elliptic curve cryptographic (ECC). Wang et al. (2006) used identity-based with bilinear map over the elliptic curves. They used an identity tree instead of key tree and divided the large group to several subgroups that each subgroup nodes were independently maintained by the Subgroup Controller (SGC). Every node could act the subgroup controller. The task of SGC was just to update the identity tree when there was a membership change. The function of session key generation and distribution was task of Key Generation Centers (KGCs). The drawback as centralized trust mechanism of this protocol presented at KGCs, if every node in subgroup was outside range of KGCs. Abdel-Hafex et al. (2004) extended authenticated two-party key agreement protocol from LQMSV protocol what was proposed by Law, Menezes, Qu, Solinas, and Vanstone (2003) to Group LMQSV (GLMQSV). This protocol also involved authentication process that was build-in key agreement process without extra communication among member nodes. They used logical ring in protocol that lead to problem because, in mobility network, the geometric of node do not correspond with logical ring.

Braid groups what was introduced by Anshel, I.; Anshel M. and Fisher (2001) has changed the concept on number theory that widely implemented in cryptographic. Several researches proposed public key cryptosystem using braid groups based on the hardness of conjugacy problem. The computation cost of braid group can decrease to number of permutation on linear algebra rather than number of exponential in traditional protocols. Ko, Lee, Cheon, Han, Kang and Park (2000) has proposed the key exchange protocol on braid groups that based on conjugacy problem in Diffie-Hellman scheme (GDH) used multiply and inverse operations within braid groups, so called Ko-Lee problem. The proposed research was different from widely used cryptosystems on number of theory, even if there are some similarities in design. Kui and Gang (2004) designed protocol on ad hoc networks based on CLIQUES protocol with dynamic operation protocol composed of join, leave, merge, partition and refresh protocol. They applied braid group cryptographic in their protocol. Most importantly, the protocol required that the members be serialized to construct the group key similar as GDH protocol.

Man-in-the-middle attack works on above protocols. The authenticated process can resistant to them. Sibert, Dehornoy and Girault (2006) introduced three authentication schemes on braid groups. The first of them was a two-pass protocol relying on a specific version of the conjugacy search problem, while the two other schemes were iterated three-pass protocols based on the conjugacy search problem and/or root problem. Shpilrain and Ushakov (2008) offered an authentication scheme whose security was based on the apparent hardness of the twisted conjugacy search problem. The suggested parameters were quite large, so that a brute force attack by exhausting the key space is not feasible. In current, there is only authentication in two-party key agreement using braid groups. Chaturvedi and Lal (2008) proposed two-pass authenticated key agreement protocol (AKAP) based on braid groups. They used long term private and public key of entities for authentication.

1.3 Concept of Purposed Protocol

The purposed protocol is developed to prevent single point of failure, to reduce computation and communication cost, to satisfy the security requirement and to prevent security attack. There are two important techniques in protocol including key tree and braid groups. The protocol is designed based on STR protocol for reducing the communication cost to constant round. The braid groups is implemented for reducing the computation cost to avoid modular exponential operation. Moreover, the purposed protocol is researched as contributory key agreement protocol in key establishment, the group members compute the group key by themselves. The security requirement is considered in all membership events including join, leave, merge, partition, and refreshing. There are two protocols in this research including the group key agreement protocol using tree-based braid groups and the extended protocol with authentication.

CHAPTER 2

LITERATURE REVIEW

There are some definitions and terminology regarding authenticated key agreement protocol. A key agreement protocol is a key establishment technique whereby a shared secret key is derived by two(or more) parties as a function of information contributed, or associated with, each of these, such that no party can predetermine the resulting value. A key agreement protocol is contributory if each party equally contributes to the key and guarantees its freshness. Let A and B be two honest parties i.e. legitimate who execute the steps of a protocol correctly. A key agreement protocol is said to provide implicit key authentication (of B to A) if the party A is assured that no other party aside from a specially identified second party B can possibly learn the value of a particular secret key. A key agreement protocol which provides implicit key authentication to both participating parties is called an authenticated key agreement protocol (A-KA). A protocol provides key confirmation if a party is assured that its peer (or a group thereof) actually has possession or a particular secret key. A contributory key agreement protocol provides key integrity if a party is assured that its particular secret key is a function of only the individual contributions of all protocol parties. In particular, extraneous contribution(s) to the group key cannot be tolerated even if it does not afford the attacker(s) with any additional knowledge. A protocol is said to have perfect forward secrecy if compromise of long-term keys does not compromise past session keys. A protocol is said to be vulnerable to known-key attack if compromise of past session keys allows either a passive adversary to compromise future session keys, or an active adversary to impersonate one of the protocol parties.

2.1 Network Security

When discussing network security, three aspects is covered; the services required, the potential attacks and the security mechanisms.

The security services aspect includes the functionality that is required to provide a secure networking environment, while the security attacks cover the methods that could be employed to break these security services. Finally the security mechanisms are the basic building blocks used to provide the security services.

2.1.1 Security Services

In providing a secure networking environment some or all of the following services may be required:

2.1.1.1 Confidentiality

To ensure that transmitted information can only be accessed by the intended receivers.

2.1.1.2 Authentication

To allow the communicating parties to be assured of the others identity.

2.1.1.3 Integrity

To ensure that the data has not been altered during transmission.

2.1.1.4 Non-repudiation

To ensure that parties can prove the transmission or reception of information by another party, i.e. a party cannot falsely deny having received or sent certain data.

2.1.1.5 Availability

To ensure that the intended network services are available to the intended parties when required. Depending on the capabilities of any potential attacker different mechanisms may be used to provide the services above.

2.1.2 Security Attacks

Security attacks can be classified in the following two categories depending on the nature of the attacker.

2.1.2.1 Passive attacks

The attacker can only eavesdrop or monitor the network traffic. Typically this is the easiest form of attack and can be performed without difficulty in many networking environments, e.g. broadcast type networks such as Ethernet and wireless networks.

2.1.2.1 Active attacks

The attacker is not only able to listen to the transmission but is also able to actively alter or obstruct it.

2.1.3 Majority of attacks

Furthermore depending on the attacker actions, the following subcategories can be used to cover the majority of attacks.

2.1.3.1 Eavesdropping

This attack is used to gain knowledge of the transmitted data. This is a passive attack which is easily performed in many networking environments as mentioned above. This attack can easily be prevented by using an encryption scheme to protect the transmitted data.

2.1.3.2 Traffic analysis

The main goal of this attack is not to gain direct knowledge about the transmitted data, but to extract information from the characteristics of the transmission, e.g. amount of data transmitted, identity of the communicating nodes etc. This information may allow the attacker to deduce sensitive information, e.g. the roles of the communicating nodes, their position etc. Unlike the previously described attacks this one is more difficult to prevent.

2.1.3.3 Impersonation

Here the attacker uses the identity of another node to gain unauthorized access to a resource or data. This attack is often used as a prerequisite to eavesdropping. By impersonating a legitimate node, the attacker can try to gain access to the encryption key used to protect the transmitted data. Once this key is known by the attacker, he can successfully perform the eavesdropping attack.

2.1.3.4 Modification

This attack modifies data during the transmission between the communicating nodes, implying that the communicating nodes do not share the same view of the transmitted data. An example could be when the transmitted data represents a financial transaction where the attacker has modified the transactions value.

2.1.3.5 Insertion

This attack involves an unauthorized party, who inserts new data claiming that it originates from a legitimate party. This attack is related to that of impersonation.

2.1.3.6 Replay

The attacker retransmits data previously transmitted by a legitimate node.

2.1.3.7 Denial of service

This active attack aims at obstructing or limiting access to a certain resource. This resource could be a specific node or services or the whole network.

2.1.4 Security Mechanisms

Most of the security services previously mentioned can be provided using different cryptographic techniques. The following subsections give an overview of which techniques are used to provide each of the services.

2.1.4.1 Confidentiality

It prevents all but those authorized from having the content of the message. There are multiple possibilities to provide confidentiality. They vary from physical to mathematical methods. For example the information to be protected, such as the key, is stored in a room that can only be accessed by the authorized users. Using some encryption algorithms the information can be encrypted so that only the user who has the proper keys and knows the algorithms is able to decrypt it. Encryption can further be roughly classified into two categories: symmetric key encryption and asymmetric key encryption.

2.1.4.2 Integrity

It prevents unauthorized users from altering data. To assure data integrity, one must be able to detect any unauthorized manipulation of data, such as deletion and insertion. One of the cryptographic approaches to achieve integrity is digital signatures. Another possibility is given by hash functions.

2.1.4.3 Non-repudiation

It prevents an entity from denying previous actions. In other words, it is a method by which the sender of data is provided with proof of delivery and the recipient is assured of the sender's identity, so that neither can later deny having processed the data. For example, someone who submits electronic order should not be able to deny it later. One of the cryptographic approaches to achieve non-repudiation is digital signatures.

2.1.4.4 Availability

The service should be available all the time. It must be robust enough to tolerate network failures and must be resistant against Denial-of-Service (DoS) attacks.

2.1.4.5 Authentication

Authentication is classified into two categories: entity authentication and data origin authentication. In the first case any party entering into a communication session must identify themselves to other participants. In the second case, sent data should be authenticated with respect to its content, time of sending, etc.

2.2 Cryptographic

A cryptographic solution to achieve confidentiality is using encryption. An encryption function maps a plain text into a chipper text (meaningless data) for given key. An encryption scheme is said to be symmetric-key if encrypting key is equal to decrypting key, it is computationally easy. Thus two parties wishing to communicate securely need to share the key over some secure channel before they can use the encryption scheme to communicate over an insecure channel. In contrast, in asymmetric-key systems it is infeasible to determine decrypting key from given encrypting key. Thus every user in such a system has a key pair both encrypting and decrypting key which is unique to them. This scheme does away with the need for a secure channel at any time. While symmetric-key techniques are much faster than asymmetric ones, they require the parties to have a pre-shared secret. Thus a common solution is to exchange a symmetric key using asymmetric technique. This holds particularly true for mobile ad hoc networks as the use of asymmetric-key cryptography for securing all communication is practically impossible.

2.2.1 Symmetric Key Encryption

Symmetric encryption is illustrated in Figure 2.1. The plain text message *m* is encrypted using the shared key *k*, resulting in the cipher text c. To recover the plain text message the cipher text is decrypted using the same key used to for the encryption. Symmetric encryption schemes can be used to provide confidentiality, integrity and authentication. The non-repudiation can provide, if it uses digital signature with Big Brother. The shared key must be distributed over a secure communication channel.

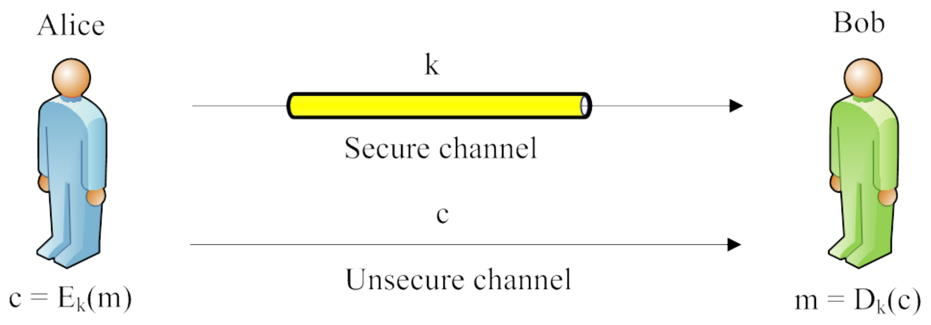


Figure 2.1 Symmetric encryption schemes

2.2.2 Asymmetric Key Encryption

Unlike conventional encryption schemes where the involved parties share a common encryption/decryption key, asymmetric key encryption schemes depend on the use of two different but mathematically related keys. One of the keys is used for encryption and the other for decryption. The asymmetric key encryption scheme is illustrated in Figure 2.2. Bob generates a pair of keys, his public/private key pair Pk_{Bob}/Sk_{Bob}. The public key is related to the private key, but in such a way that the private key cannot be derived from it without additional information.

If Alice wants to send an encrypted message to Bob, she first needs to obtain his public key. As the name implies Bob's public key does not need to be kept secret, however it must be authenticated, i.e. Alice must be assured that the public key she believes belongs to Bob is really his.

Once Alice has Bob's authentic public key Pk_{Bob}, she encrypts the plain text message *m* using it. The resulting cipher text *c* can then only be decrypted using Bob's private key Sk_{Bob} which only Bob knows.

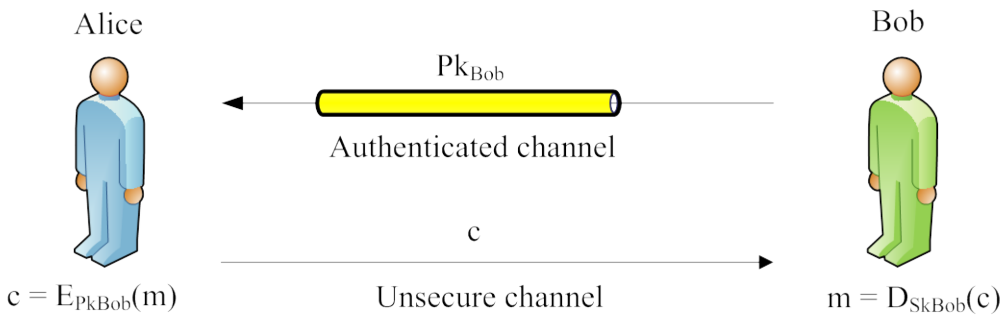


Figure 2.2 Asymmetric key encryption schemes

Compared with symmetric key encryption, asymmetric key encryption has a weaker requirement for the communication channel over which the key distribution is performed. Asymmetric key encryption only requires an authenticated channel as opposed to a secure channel that is required for the distribution of symmetric encryption keys. Asymmetric key encryption can also provide non-repudiation along with confidentiality, integrity and authentication. However, asymmetric key encryption requires much more computational resources than symmetric encryption and therefore has much lower performance. Therefore public key encryption is typically only used to encrypt small amounts of data, e.g. symmetric encryption keys and digital signatures.

2.2.3 Digital Signature

A digital signature is a data structure that provides proof of origin, i.e. authentication and integrity, and depending on how it is used, it can also provide non-repudiation. Figure 2.3 illustrates how a digital signature is used. Alice wants to send a message to Bob, however she doesn't want it to be modified during transmission and Bob wants to be sure that the message really came from Alice. What Alice does is that she computes a hash digest of the message which she encrypts with her private key Sk_{Alice}. She then sends both the message and the encrypted digest which is here signature. Bob can then verify the signature by computing the hash digest of the message he received and comparing it with the digest he gets when decrypting the signature using Alice's public key Pk_{Alice}. If the digests are equal Bob knows that Alice sent the message and that it has not been modified since she signed it.

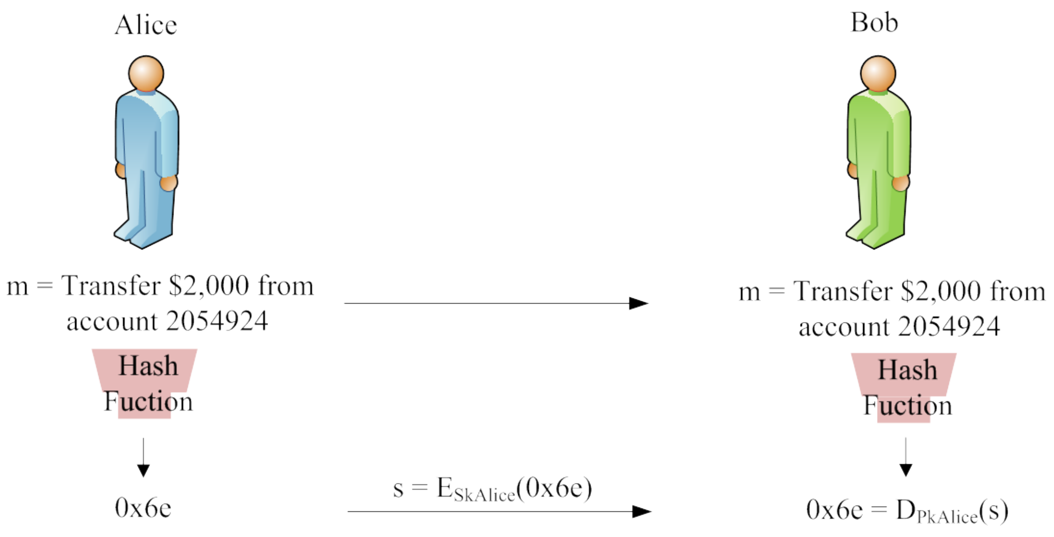

Alice
Bob
m = Transfer $2,000 from account 2054924
m = Transfer $2,000 from account 2054924
Hash Fuction
Hash Fuction
0x6e
s = E_SkAlice(0x6e)
0x6e = D_PkAlice(s)

Figure 2.3 Example of a digital signature

2.2.4 Key Establishment

Key establishment may be broadly divided into key transport and key agreement. A key transport protocol or mechanism is a key establishment technique where one party creates or otherwise obtains a secret value, and securely transfers it to the other(s). A key agreement protocol or mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these, (ideally) such that no party can predetermine the resulting value. Thus in scenarios, where there is no central authority and the task of key generation cannot be assigned to a single (or few) participant(s)(as is most often the case in ad hoc networks), key agreement is a good alternative to key transport. But more often in a key agreement protocol, it is required that a participating member be assured that no other party aside from a specially identified party (or parties) may gain access to a particular secret key. Such a protocol is known as an authenticated key agreement protocol. It is worth noting here that the term authentication, in this context, means to have the knowledge of the identity of parties who may gain access to the key. To corroborate the fact that the same identities actually participated in the protocol, one has to rely on other mechanisms like entity authentication. Key agreement protocols can be designed using symmetric or asymmetric techniques. To further illustrate the above concepts present the two-party Diffie-Hellman key agreement protocol.

2.2.4.1 The Diffie-Hellman Key Agreement

Developed by Diffie and Hellman, this algorithm allows the establishment of a cryptographic secret key between two entities by means of data exchange through an insecure communication channel. The algorithm executed between two entities A and B is defined as follows:

1) A and B agree on two randomly chosen numbers p and g, so that p is a large prime number and $g < p$;

2) A chooses a secret random number S_A and B chooses a secret random number S_B;

3) A computes a public value $T_A = g^{S_A}$ mod p and B computes a public value $T_B = g^{S_B}$ mod p;

4) A sends T_A to B and B sends T_B to A;

5) A computes $T_B{}^{S_A}$ mod $p = (g^{S_B})^{S_A}$ mod p and B computes $T_A{}^{S_B}$ mod $p = (g^{S_A})^{S_B}$ mod p.

Since $(g^{S_A})^{S_B}$ mod $p = (g^{S_B})^{S_A}$ mod $p = K$, these two entities share a secret cryptographic key K.

In other works, two entities are able to exchange information through a channel that anyone can listen to and at the end of the process the two entities, and only the two entities, share the same secret key.

The security of this algorithm is based on the difficulty to calculate the secret key $K = g^{S_A S_B}$ mod p, despite of knowing the public values g^{S_A} mod p and g^{S_B} mod p, when the prime number p is sufficiently large.

This algorithm has a weakness that consists on the lack of authentication between the two entities. Even though they are able to establish a secret key, there is no guarantee that these entities are who they claim to be. This weakness is man-in-the-middle attack problem.

2.2.4.2 Group Key Agreement

Key agreement protocols for more than two parties are known as Group Key Agreement (GKA) protocols. These get a bit involved as the two-party Diffie-Hellman key agreement does not extend trivially to more than two parties. They are well suited to the security needs of small, dynamic, collaborative groups as they offer the possibility of creating session keys for each group session, thus adjusting well to group membership changes. Key features of a GKA (Group Key Agreement) protocol are:

1) Contributory

GKA protocols are contributory in nature which means that all members participating in the protocol contribute towards the secret and even in the absence of one contribution it is infeasible to derive the secret.

2) Lack of a central authority

There is no single member controlling the execution of the protocol. Even if there is some “leader” it is short-lived and restricted to that particular execution of the protocol.

3) Key Freshness

The secret key derived cannot be predicted in advance (even by one of the protocol participants).

The key derived from the GKA protocol needs to meet the following security features:

1) Group key secrecy

It simply means that the derived secret should not be derivable by a non-participant.

2) Forward key secrecy

Merely knowing one of the current group keys, one should not be able to compute previous group keys.

3) Backward key secrecy

Merely knowing one of the current group keys, one should not be able to compute future group keys.

4) Key independence

Knowledge of a subset of group keys in the life-cycle of a group (by a participant or outsider) should not enable knowledge of any key outside this subset.

5) Perfect forward secrecy

Knowledge of a long-term secret should not enable one to compute past group keys.

Thus in the context of ad hoc networks, GKA protocols provide a mechanism to derive a (symmetric) group key in the absence of a third party.

2.3 Group Key Agreement Protocol

Group Key Agreement protocols find applications in many group applications including telephone and video conferences, remote consultation and diagnosis systems for medical applications, contract negotiation, multi-party games, collaborative work places, electronic commerce environments such as on-line real-time auctions, and information dissemination of stock quotes. Many GKA protocols have been proposed in the literature. While some are only suitable for static groups others work in case of certain kinds of groups only (for instance groups with certain number of members or groups with the ability to listen to multiple broadcasts in a single round). While security flaws have been found in some others. This study presents here protocols which work in case of dynamic groups and are very generic in their assumptions about the group sizes and dynamics and with no known security flaws in them. Each protocol is defined in terms of the following operations:

• Initial Key Agreement (IKA): This refers to the setup stage when a number of new users decide to derive a new group key.

• Auxiliary Key Agreement (AKA): This refers to the group modification procedures (after a group is formed). It is essential that each of these operations lead to a change in the group key (to maintain key independence). These operations are:

Join: One new member wishes to join an already established group.

Delete: A member leaves (voluntarily or otherwise) a group.

Partition: A group is divided into two (or more) smaller groups. It can also be viewed as a "Delete" of more than one member.

Merge: Two (or more) groups get together to form a single group. It can also be viewed as a "Join" by more than one member.

2.3.1 The Burmester and Desmedt Protocol

This protocol was presented by Burmester and Desmedt (1994) and was executed in three rounds. Each participant M_i, $i \in [1, n]$ executes the following operations:

1) To generate a secret random value r_i and broadcasts $Z_i = g^{r_i} \bmod p$ to the other participants;

2) To compute and broadcasts $X_i=(Z_{i+1}/Z_{i-1})^{r_i}$ to the other participants;

3) To compute the group key $K_n=Z_{i-1}^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdot \ldots \cdot X_{i+(n-2)}$

This group key has the form $K_n= g^{r_1r_2+r_2r_3+\ldots+r_{n-1}r_n} \bmod p$ and shares the security characteristics presented by the Diffie-Hellman algorithm. This protocol is efficient with respect to the total number of rounds. This characteristic could allow faster execution, but each round requires n simultaneous broadcasts. Simultaneous broadcasts are usually not possible, even in wireless networks, because there can be only one broadcast message at a given moment. Due to this characteristic, the deployment of this protocol must use sequential broadcast messages. Since each broadcast message acts like a round, there is no longer a low number of rounds advantage. Another disadvantage is that this protocol makes use of a high number of exponential operations.

2.3.2 The Hypercube and Octopus Protocols

The Hypercube protocol was presented by Becker and Willie (1998) and intent to overcome the high number of messages needed by logically arranging the nodes in a hypercube. For a network consisting of four nodes positioned as a square, a key is established between A and B (i.e. $g^{S_aS_b}$), and another key between C and D (i.e. $g^{S_cS_d}$). These keys are used to establish a single key, $g^{g^{S_aS_b} \cdot g^{S_cS_d}}$, among the four entities, as presented in Figure 2.4. This behavior can be generalized for higher numbers of nodes, as long as the number of participants equals 2^d, for $d \in$ I. This protocol executes in *d* rounds.

The Octopus protocol is an extension of the Hypercube protocol for networks with an arbitrary number of nodes. A subgroup of nodes is arranged in a hypercube, composing a core. Each core node establishes a key with each nearby non-core node using the Diffie-Hellman protocol. The product of these keys is used to establish a key among the core nodes as specified by the Hypercube protocol. This key is then distributed to the other nodes.

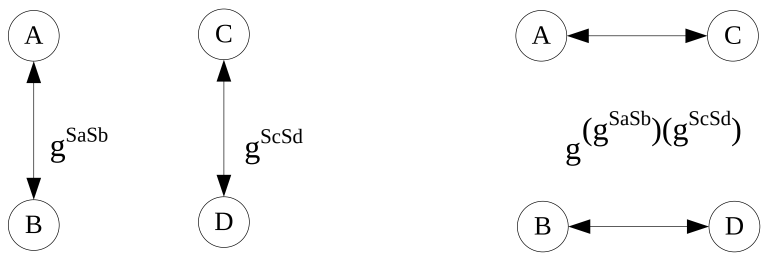


Figure 2.4 Hypercube protocol for n = 4

2.3.3 The CLIQUES Protocol Suite

Developed by Steiner et al. (1998), the CLIQUES protocol suite consists of key management protocols for dynamic groups. Two of these protocols, IKA.1 and IKA.2 (Initial Key Agreement 1 and 2), are defined for group key establishment. Other protocols are specified for member and subgroup addition and exclusion and key refresh. The protocols from this suite can provide member authentication, which solves the Diffie-Hellman authentication vulnerability.

2.3.3.1 IKA.1 Protocol

The IKA.1 protocol executes in two stages:

1) $M_i \rightarrow M_{i+1}$:

$$\{g^{\frac{S_1 S_2 \cdots S_i}{S_k}} \mid k \in [1, i]\}, g^{S_1 S_2 \cdots S_i}$$

2) $M_n \rightarrow M_i$:

$$\{ g^{\frac{S_1 S_2 \cdots S_n}{S_i}} \mid \mathrm{i} \in [1, n - 1] \},$$

for $i \in [1, n - 1]$.

At the first stage contributions are collected from all group members throughout n - 1 rounds. Each group member (except the first) receives a data set that represents the partial contributions from all the group members that have already executed this first stage. The member adds its contribution and sends a new data set to the next group member.

The last group member, M_n, is called the group controller. At the end of the first stage it receives a data set whose cardinal value is $g^{S_1 S_2 \cdots S_{n-1}}$ and computes the group key $K_n = g^{S_1 S_2 \cdots S_n}$.

At the second stage, the group controller adds its contribution to each intermediate value and broadcasts this new data set to every other node in the network. Each intermediate value now consists of the contribution of all group members except one. In order to compute the group key, each group member M_i identifies the appropriate intermediate value (the one that does not contain its contribution) and raises it to its contribution S_i, obtaining K_n.

2.3.3.2 IKA.2 Protocol

The IKA.1 protocol requires i + 1 exponential operations when executed by the i^{th} node. In some environments it is desirable to minimize the computational effort demanded from each group member. Some examples of these environments are groups with a high number of members and groups whose members have limited computational capacity. The IKA.2 protocol was proposed in order to minimize the demanded computational cost. It is similar to the IKA.1 protocol but is executed in four stages:

1) $M_i \rightarrow M_{i+1}$; $i \in [1, n - 2]$:

$$g^{S_1 S_2 \cdots S_i}$$

2) $M_{n-1} \rightarrow M_i$; $i \in [1, n - 2]$:

$$g^{S_1 S_2 \cdots S_{n-1}}$$

3) $M_i \rightarrow M_n$; $i \in [1, n - 1]$:

$$g^{\frac{S_1 S_2 \cdots S_{n-1}}{S_i}}$$

4) $M_n \rightarrow M_i$; $i \in [1, n - 1]$:

$$\{ g^{\frac{S_1 S_2 \cdots S_n}{S_i}} \mid i \in [1, n - 1] \}.$$

At the first stage contributions are collected from the n-2 first group members by means of a single message sent from one member to the next that gathers all the previous contributions. At the second stage, M_{n-1} adds its contribution to the received message and broadcasts this new message to the n - 2 first members. At the third stage each member factors out its own contribution and sends this result to the last group member. At the last stage, M_n collects all the sets from the previous stage, raises each one of them to its contribution S_n and broadcasts these results to the other group members, allowing them to compute the group key.

These two protocols have the advantage of requiring a low number of messages. The IKA.2 protocol has reduced the number of exponential operations required for group key establishment. Unlike the other presented protocols, this protocol suite provides mechanisms for group addition and exclusion, making it unnecessary to execute the entire key establishment protocol. This characteristic reduces the involved costs and provides backward and forward secrecy.

2.3.3.3 Join Protocol

To handle a join event, a group controller is needed. It can be any member in the group. The group controller has usually higher performance capability than other members. Let M_c be the group controller, and let M_{i+1} be the potential member. In first step, M_c chooses a new random secret exponent S_c. The message will be sent by M_c to the new member M_{i+1}. M_{i+1} generates randomly its secret S_{i+1} and embeds S_{i+1} in the message which it then broadcasts to all members in group. After running join protocol, all members compute the new group key.

2.3.3.4 Leave Protocol

The leave protocol in CLIQUES is relatively simple. It needs only one round. Let M_c be group controller, and M_l be the member which leaves the group. First, M_c updates its secret S_c. M_c constructs then a broadcast message by embedding its new secret. Finally M_c broadcasts the message to the group. After running leave protocol, all remaining members compute the new group key. Although the contribution S_l is still factored into the new group key, the left member M_l is unable to compute the new group key, due to the absence of the subkey $g^{\frac{S_c S_1 S_2 \cdots S_n}{S_l}}$.

2.3.4 Tree-based Group Key Agreement (TGDH) Protocol

TGDH is an adaptation of key trees (Kim et al. 2000;2004) in the context of fully distributed, contributory group key agreement. TGDH computes a group key derived from the contributions of all group members using binary tree.

In a TGDH key tree, the node located at level 0, node (0, 0), is the root. The height *h* of a tree equals the shallowest level. A member is associated with a leaf node. A non-leaf node is called an internal node with two children. The left child of node (l, v) is node $(l + 1, 2v)$, and the right one is node $(l + 1, 2v + 1)$. A node (l, v) is associated with a key $K_{(l,v)}$ and the corresponding blinded key $BK_{(l,v)} = g^{K_{(l,v)}} \bmod p$. The key $K_{(l,v)}$ can be recursively computed as follows:

$$\begin{aligned} K_{(l,v)} &= BK_{(l+1,2v)}^{K_{(l+1,2v+1)}} \bmod p \\ &= BK_{(l+1,2v+1)}^{K_{(l+1,2v)}} \bmod p \\ &= g^{K_{(l+1,2v)}K_{(l+1,2v+1)}} \bmod p \end{aligned}$$

Figure 2.5 gives an example of a key tree of a group with six members. The group key for this group is

$K_{(0,0)} = g^{g^{S_3 \cdot g^{S_1 \cdot S_2}} \cdot g^{S_6 \cdot g^{S_4 \cdot S_5}}} \bmod p$, where $S_1, \cdots, S_6$ are the keys of members $M_1, \cdots, M_6$ respectively.

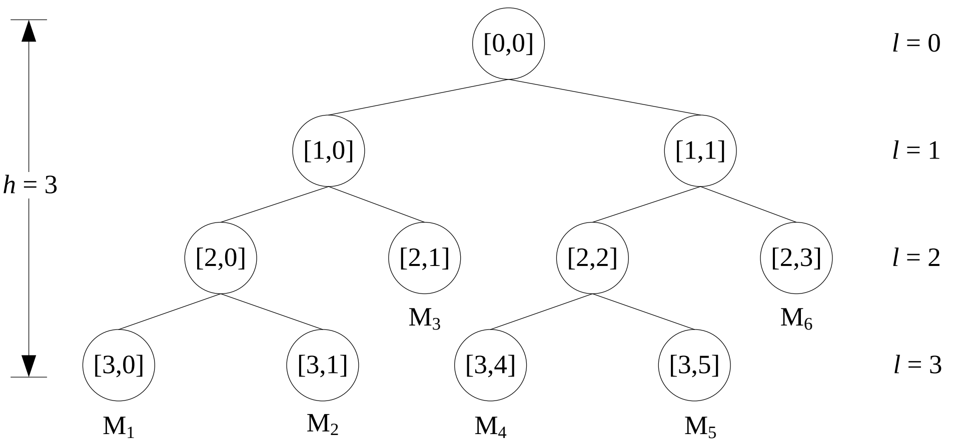


Figure 2.5 TGDH tree

The overhead of the TGDH protocol depends on many factors, e.g. tree height, balance of the key tree, location of insertion points and leaving members. Hence some characteristics can only be estimated for the worst case, and some issues can even not be estimated.

Assume that it has a TGDH tree with n members, there are $n-1$ internal nodes. Each member must know all blinded keys of all nodes except for the root, and all keys along the path from that member node to the root node.

The rest of this section gives an overview of the protocols setup, join, leave, merge and partition, and refresh in TGDH.

2.3.4.1 Join protocol

Assume that a new member M_{n+1} joins to a group of n members $\{ M_1, \cdot\cdot\cdot, M_n \}$. The new member broadcasts its join-request with its own BK_{n+1}. After receiving the join-request, each member in current group determines the insertion point. If the tree is fully balanced, the new member joins to the root node. Otherwise, the shallowest rightmost leaf node is the insertion point. The reason of such selection is to keep the key tree as balanced as possible.

The tree must be first updated. The sponsor is the shallowest rightmost leaf in the subtree rooted at the insertion node. A new intermediate node is created. Node at the insertion point becomes the left child of the new intermediate node, and the new member becomes the right child. The process of join protocol in TGDH is illustrated as follows:

1) The new member broadcasts its join-request with its blinded key.

2) Every member updates key tree, removes all keys and blinded keys from sponsor to the root node. The sponsor M_s additionally, updates its share, computes then all keys and blinded keys in its key-path, and broadcasts the updated tree with all blinded keys.

3) Every member computes the group key using new key tree.

Figure 2.6 gives an example of a group with three members when a new member M_4 joins into this group. Member M_3 is selected as the sponsor. Each member in the old tree updates the tree by inserting M_4 and removes the keys and blinded keys in M_3's key-path. The sponsor M_3 additionally updates its share and computes $K_{(1,1)}$, $BK_{(1,1)}$, and $K_{(0,0)}$, and then broadcasts the updated tree with all blinded keys (i.e. $BK_{(2,0)}$, $BK_{(2,1)}$, $BK_{(2,2)}$, $BK_{(2,3)}$, $BK_{(1,0)}$ and $BK_{(1,1)}$). Equipped with the broadcast message, M_1 and M_2 computes the new $K_{(0,0)}$, and M_4 computes $K_{(1,1)}$ and $K_{(0,0)}$.

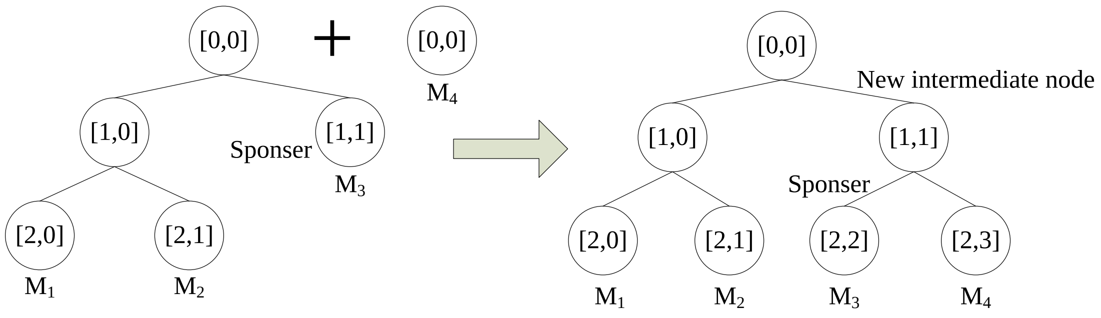


Figure 2.6 An example of TGDH join

2.3.4.2 Leave Protocol

The leave protocol is relatively simple. Assume that it has a group of n members and member M_l leaves the group. The rightmost leaf node of M_l's sibling subtree is selected as the sponsor. After notification from system about the leave event, each remaining member updates its key tree by deleting M_l. The former sibling of M_d is promoted to replace M_d's parent node. The sponsor must additionally refresh keys in its key-path. The process of leave protocol is illustrated as follows:

1) Each member removes the leaving member and relevant parent node, removes all keys and blinded keys pairs from sponsor to root node. The Sponsor M_s additionally, updates its share, computes then all keys and blinded keys in its key-path, and broadcasts updated key tree including all blinded keys.

2) Every member computes the group key using new key tree.

Figure 2.7 gives an example of a group of five members when M_3 leaves this group. M_5 is selected as the sponsor. The remaining members M_1, M_2, M_4, and M_5 remove the nodes (2,2) and (1,1). M_5 additionally updates its share and computes $K_{(1,1)}$, $BK_{(1,1)}$, $K_{(0,0)}$, and then broadcasts the updated tree with all blinded keys. Equipped with this broadcast message, M_1 and M_2 compute new $K_{(0,0)}$, and M_4 computes $K_{(1,1)}$, and $K_{(0,0)}$.

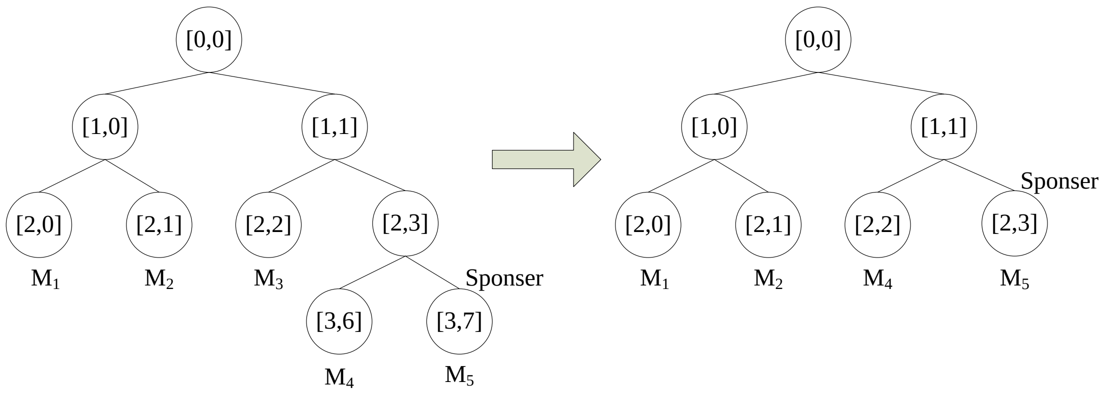


Figure 2.7 An example of TGDH leave

2.3.4.3 Merge Protocol

Compared to CLIQUES, the main virtue of TGDH is that it is much simpler to merge two or more groups. Multiple join can be processed as follows. The protocol assumes that *m* members want to join group G_1. The *m* individual members form a TGDH group G_2. Then G_2 merges with G_1. The protocol considers first the merge of two groups. It can be simply extended to the merge of more than two groups, say $k > 2$, groups by executing the two-group merge $k - 1$ times.

First the two trees are ordered from the highest to lowest, denoted T_1 and T_2. If they are of the same height, they are ordered according to some other criteria. T_2 joins to T_1, and the insertion point is determined. If the two trees are of the same height, it joins simply T_2 to the root node of T_1. Otherwise it first tries to find the rightmost shallowest node where the join would not increase the overall tree height. If no such node exists, the insertion point is the root node.

Assume that we have m trees to be merged. They can be ordered from the highest to the lowest: $T_1, \cdot\cdot\cdot, T_m$. To perform merge, each tree T_i has its rightmost shallowest leaf node as sponsor Ms_i . The process is illustrated as follows:

1) Each sponsor Ms_i in tree T_i updates its share, computes all keys and blinded keys in the key-path of T_i (including $BK_{(0,0)}$), broadcasts updated tree T_i including only all blinded keys. For $i = 1, 2, \cdot\cdot\cdot, m$. Each member additionally, updates the key tree and determinates the new sponsors $Ms_1, \cdot\cdot\cdot, Ms_m$, removes all keys and blinded keys in sponsors' key-paths.

2) To repeat this step until any sponsor computes group key. Each sponsor Ms_i with $1 \le i \le m$, computes all keys and blinded keys pairs in the key-path as far as possible, and broadcasts updated tree with all blinded keys.

3) Every member computes the group key new key tree.

An example of the merge of two groups is given in Figure 2.8. Both sponsors M_5 and M_7 first update their shares respectively. Then M_5 computes new $K_{(1,1)}$, $BK_{(1,1)}$, $K_{(0,0)}$, and $BK_{(0,0)}$ in tree T_5, and M_7 computes $K_{(0,0)}$, and $BK_{(0,0)}$ in tree T_7. Both sponsors M_5 and M_7 broadcast their updated trees with all blinded keys. Each member merges both trees independently, and chooses M_2 as the new sponsor. All members then remove all keys and blinded keys in M_2's key-path. M_2 additionally updates its share and computes new $K_{(2,1)}$, $BK_{(2,1)}$, $K_{(1,0)}$, $BK_{(1,0)}$, and $K_{(0,0)}$. M_2 then broadcasts the updated tree with all blinded keys. Equipped with the broadcast message, M_1 computes $K_{(2,0)}$, $K_{(1,0)}$, and $K_{(0,0)}$. M_6 and M_7 compute $K_{(1,0)}$, and $K_{(0,0)}$. All other members M_3, M_4, and M_5 compute only the new group key $K_{(0,0)}$. Since there is only one sponsor for the merge of two groups, M_2 knows all blinded keys in its co-path so that it is able to compute all keys and blinded keys in its key-path.

2.3.4.4 Partition Protocol

Assume that it has a group of *n* members and *k* of them leave the group. In the first round, every remaining member updates its tree by deleting all partitioned members as well as their respective parent nodes. In other words, if all leaf nodes of a subtree leave the group, the root node of this subtree is marked as leaving (namely the whole subtree is marked as leaving) and its leaf nodes are removed from the leaving nodes list. For each leaving node it identifies a sponsor using the same criteria as described in Section 2.3.4.2. The process of partition protocol is illustrated as follows:

1) Every member updates key tree by deleting all leaving member nodes and their parent nodes, removes all keys and blinded keys from sponsors to the root node. The shallowest rightmost sponsor additionally updates its share.

2) To repeat this step until any sponsor computes the group key. Each sponsor Ms_i computes all keys and blinded keys in the key-path as far as possible, broadcasts updated tree including all blinded keys.

3) Every member computes the group key using new key tree

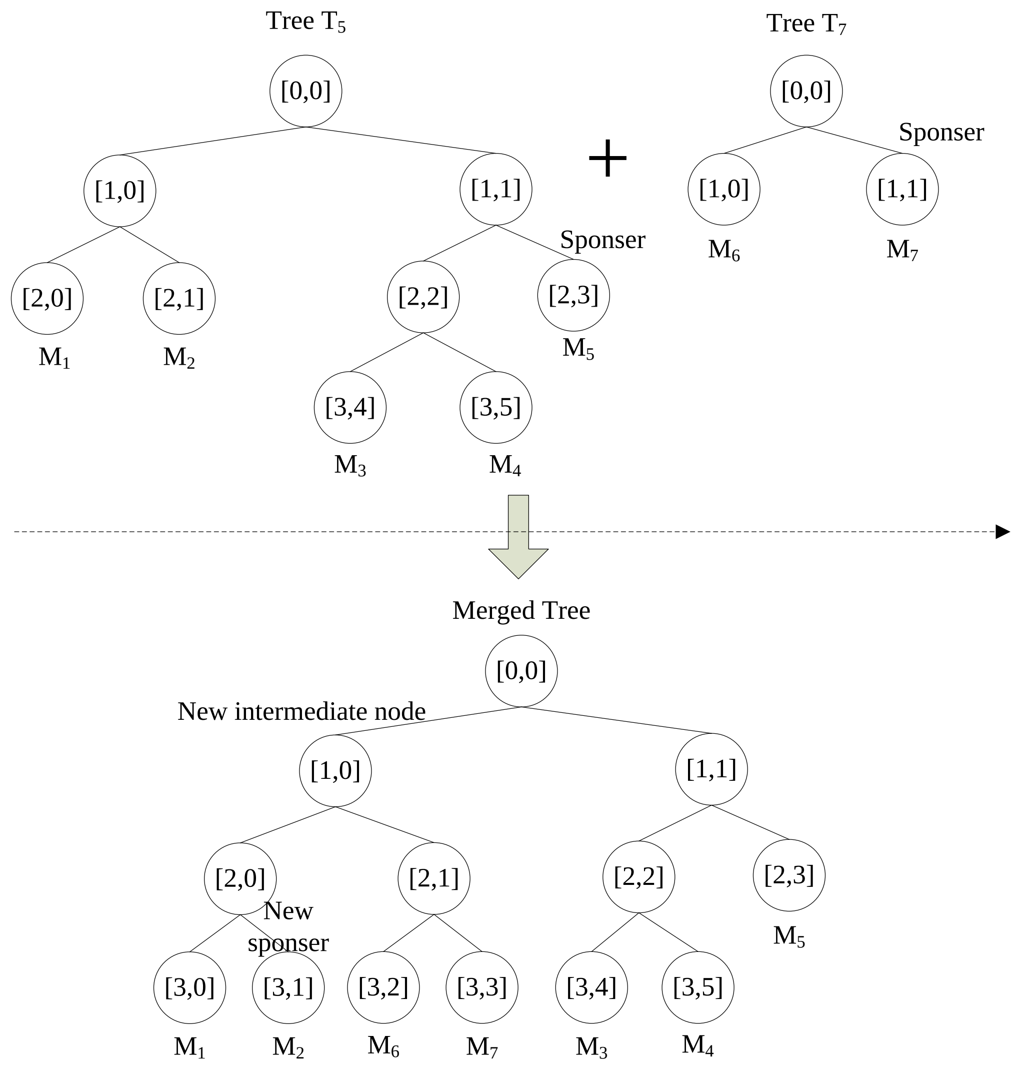


Figure 2.8 An example of 2-group TGDH merge

An example of TGDH partition is given in Figure 2.9. A group of seven members M_1, · , M_7 is partitioned. From the perspective of M_5, members M_2, M_4, M_6 and M_7 leave the group, so M_5 is in tree T_5 after the partition, the same for M_1 and M_3. However from the perspective of M_4, the leaving members are M_1, M_3, and M_5. So M_4 is in tree T_4 after the partition, the same manner is for M_2, M_6, and M_7.

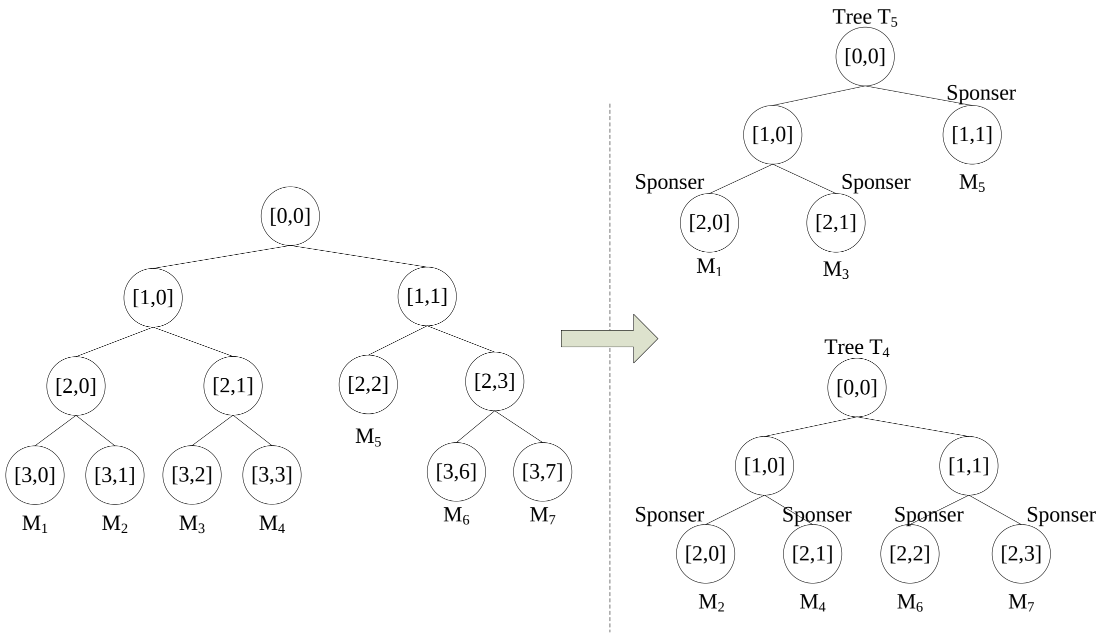


Figure 2.9 An example of TGDH partition

2.3.5 STR Protocol

STR is basically an “extreme” version of TGDH, where the key tree structure is completely imbalanced or stretched out. This protocol and its features are described in details (Steer et al.,1998; Kim et al, 2001). Like TGDH, the STR protocol uses a tree structure that associates the leaves with individual random session contributions of the group members. Every internal (non-leaf) node has an associated secret key and a public blinded key. The secret key is the result of a Diffie-Hellman key agreement between the node’s two children. The group key is the key associated with the root node.

STR uses a key tree to manage the key group. The tree has two types of nodes, namely leaf and internal nodes. Each specific group member M_i is associated with a leaf node LN_i, while an internal node IN_i has two children: the left child IN_{i-1}, and the right child LN_i. Each leaf node LN_i generates randomly a session random s_i which should be kept secretly, and computes the corresponding blinded session random

$br_i = g^{s_i}$. An internal node IN_i has a secret key k_i and the corresponding public blinded key $bk_i = g^{k_i}$. The difference is that k_i is not randomly chosen, but the result of a two-party DH key exchange between its two children:

$$k_i = bk_{i-1}^{s_i} = br_i^{k_{i-1}} \equiv g^{k_{i-1}s_i} \bmod p,\ i > 1. \qquad (2.1)$$

The internal node with the greatest index is considered to be the root. The secret key of the root is the shared group key. For a group of n members, the root is IN_n, and the group key k_n can be recursively computed using Equation 2.1. If $n = 4$, its group key is

$$K_G = g^{s_4 \cdot g^{s_3 \cdot g^{s_2 s_1}}} \bmod p.$$

Looking at the STR key tree with n members in Figure 2.10, the member M_c, $1 \le i \le$ n, must know its own session random, all blinded keys and blinded session random, and keys of the path from its parent node to root node. More formally, it must store s_c, br_i, bk_i for $i = 1, \cdots, n$ ($br_1 = bk_1$), and k_i for $i = c, \cdots, n$. The rest of this section describes how STR deals with the group operations

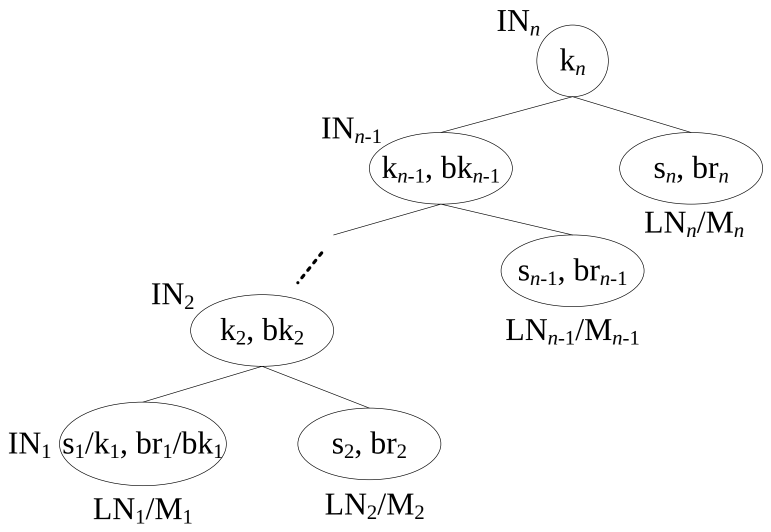


Figure 2.10 STR tree

2.3.5.1 Join Protocol

The current group has n members, the new member is identified with M_{n+1}. The tree will be updated by incrementing $n = n + 1$ and adding a new internal node IN_n with two children: the root node IN_{n-1} of the prior tree T_i on the left and the new leaf node LN_n on the right. This node becomes the new root node. Figure 2.11 gives an example of addition of a new member to a group with four members.

For simplicity, it uses n in the following to denote the number of group members before operation join. To deal with the join operation, the member M_n is chosen as the sponsor. The new member M_{n+1} broadcasts a join request containing its own blinded key bk_{n+1}. All members M_i with $1 \leq i \leq n$ can compute the new group key. The sponsor unicasts all blinded keys to M_{n+1}. Equipped with this message the new member can also compute the new group key.

However this join protocol does not provide key independence since knowledge of a previous group key can be used to compute the new group key. To remedy the situation, they suggest that the sponsor updates its session random. The changed information will then be broadcasted to all members. The process is illustrated as follows:

1) The new member broadcasts its own blinded session random br_{n+1}.

2) The sponsor $M_s(s = n)$ updates its session random s_s and br_s, computes new k_n, bk_n, and broadcasts the updated tree with all blinded keys and blinded session random.

3) Every member M_i updates the tree by inserting M_{n+1}, sets $n = n + 1$, and computes the keys:

- if $i = 1, \cdot \cdot \cdot, n - 2$, computes $k_j = br_j^{k_{j-1}}$ for $j = n - 1$,
- if $i = n$ (new member), computes $k_n = bk_{n-1}^{s_n}$,
- if $i = s$ (sponsor), computes $k_n = br_n^{k_{n-1}}$.

Figure 2.11 shows the example of M_5 join the existing group.

2.3.5.2 Leave Protocol

Like in CLIQUES, the leave protocol in STR is relatively simple, only one round is needed. Suppose it has a group of n members and the member M_l with $1 \leq l \leq n$ leaves the group. Again it needs a sponsor M_s to update its session random. If $l > 1$, the sponsor is the leaf node directly below the leaving member, i.e. M_{l-1}, otherwise the sponsor is M_2. Since the tree will be updated and renumbered (see below), if M_1 leaves the group, the sponsor is also M_1 after renumbering.

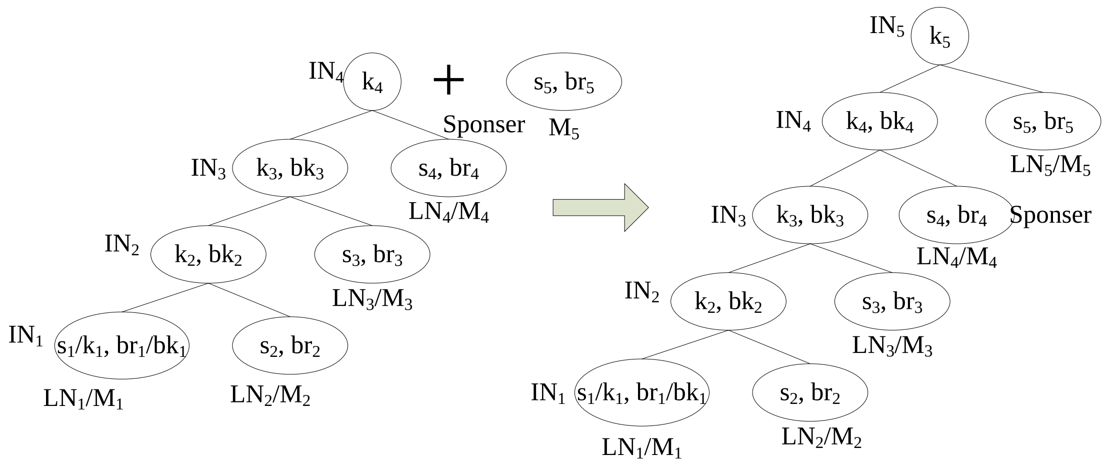


Figure 2.11 An example of STR join

After notification of the leave event from the group communication system, each remaining member updates the key tree by deleting the nodes LN$_l$ and IN$_l$, and then renumbers the nodes above M$_l$. The process is illustrated as follows:

1) Every member updates the tree by removing the leaving member M$_l$, sets $n = n - 1$.

2) The sponsor M$_s$ additionally updates its session random s$_s$ and br$_s$, computes k$_n$ and k$_i$ and bk$_i$, $\forall i \in [\max(2, s), n - 1]$, and broadcasts the updated tree with all blinded keys and br$_s$.

3) Every member M$_i$, computes the group key.

Figure 2.12 gives an example of the exclusion of a member from a group of four members.

2.3.5.3 Merge Protocol

The key tree allows relatively simple merge of two groups. The merge protocol covers also the multiple join.

The smaller group is merged onto the larger one, i.e. to place a smaller key tree directly on top of the larger one. If group sizes are equal, it can order them according to some other criteria. A new intermediate node *N* with two children is created. The root of the larger tree becomes the left child of *N*, while the lowest-numbered leaf of the smaller trees becomes the right child of *N*. The root of smaller tree becomes the root of the new tree.

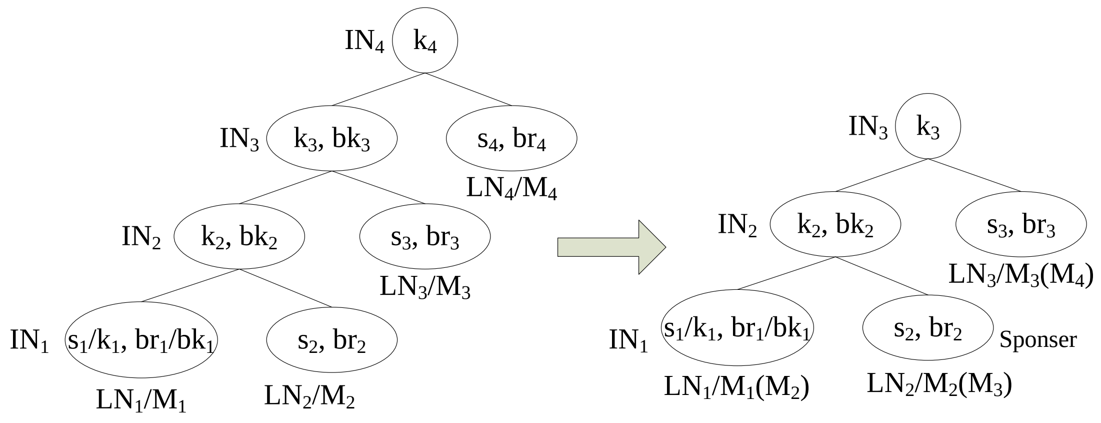


Figure 2.12 An example of STR leave

It needs a sponsor for each group, the same as in join protocol, the topmost leaf node is selected as sponsor. Both sponsors exchange key their respective key trees containing all blinded keys in the first round, the sponsor of the larger key tree becomes then the new sponsor in round 2. The new sponsor updates first its session random, and then computes all key and blinded key pair up to the new root node. It then broadcasts the new tree with all blinded keys and blinded session random.

Figure 2.13 gives an example of the merge of two groups, with three and two members respectively.

Assume It has two trees, the larger one T^1 with n_1 members, and the less one T^2 with n_2 members. The sponsors of both trees are denoted by M_{s1} and M_{s2} respectively. The process to merge two groups is illustrated as follows:

1) Both sponsors M_{s1} and M_{s2} exchange T^1 and T^2 with all blinded keys and blinded session random respectively

2) The sponsor M_s (formerly M_{s1}) updates its session random, computes (k_i, bk_i), $i = n_1 - 1, \cdot\cdot\cdot, n_1 + n_2 - 1$, and k_{n1+n2} , and broadcasts the updated tree with all blinded keys.

3) Every member M_i computes the group key.

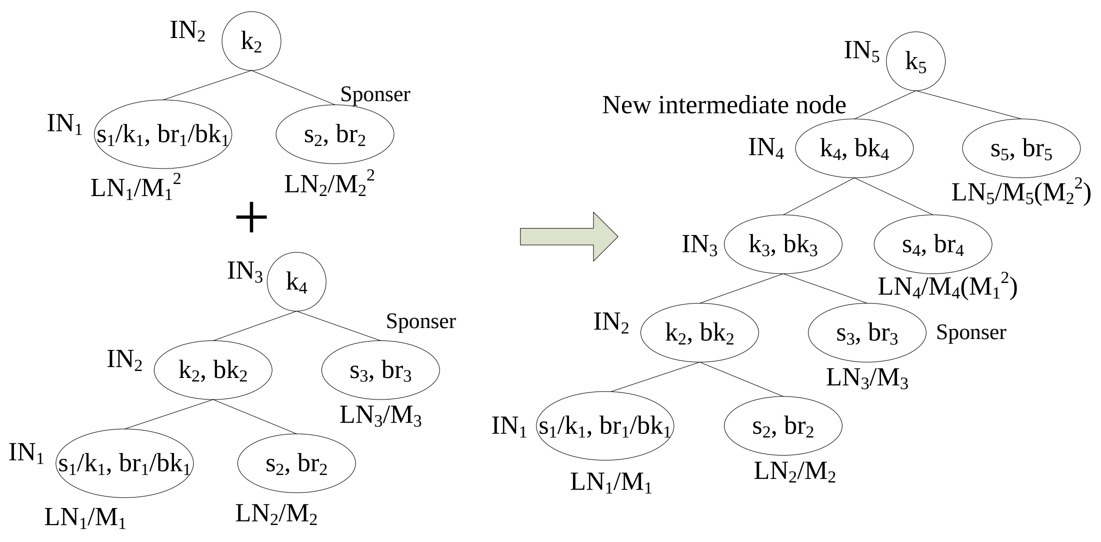


Figure 2.13 An example of STR merge

The merge protocol provides backward secrecy since all members are only given blinded keys of the other groups. However, the merge protocol does not provide key independence, since knowledge of a group key of tree T^1 used before merge can be used to compute the group key used after the merge. This problem will be remedied, if the sponsor in the second round updates its session random.

2.3.5.4 Partition Protocol

The partition protocol is similar to the leave protocol. The only difference is the choice of the sponsor. They usually choose the surviving leaf node directly below the lowest-numbered leaving member. If no such leaf node exists, in other words, if M_1 leaves the group, they choose the lowest-numbered surviving leaf node as sponsor. An example is given in Figure 2.14.

Suppose there are a group of n members when p of them leave the group. The process is illustrated as follows:

1) Every member updates the tree by removing the leaving members, sets $n = n - p$.

2) The sponsor M_s additionally updates its session random s_s and the blinded one br_s, computes new k_n, and (k_i, bk_i), $\forall i \in [\max(2, s), n - 1]$, and broadcasts the updated tree with all blinded keys and its new blinded session random.

3) Every member M_i computes the group key.

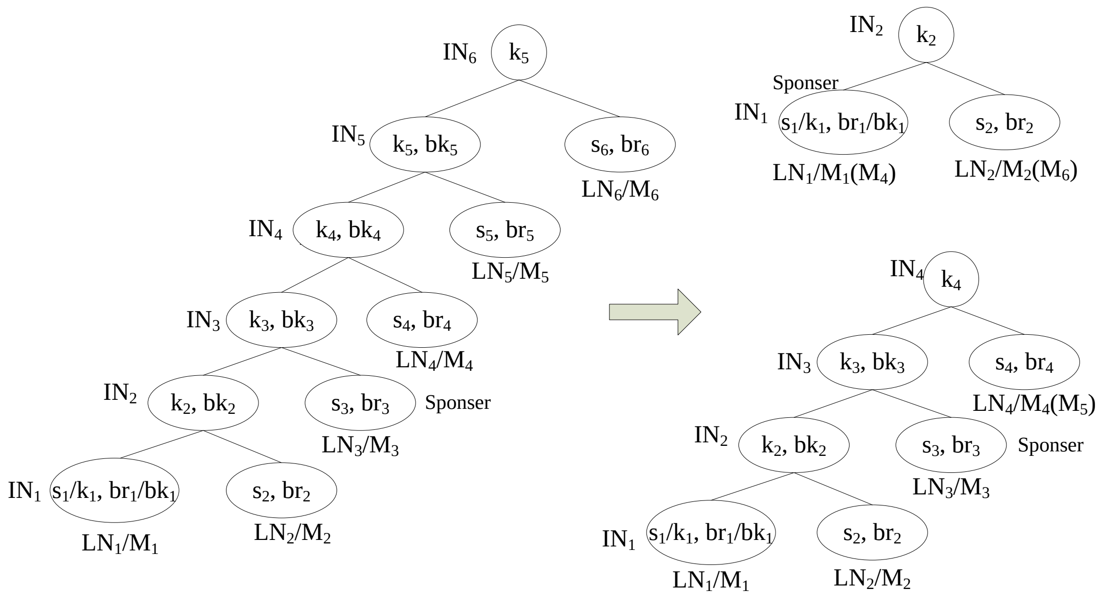


Figure 2.14 An example of STR partition

2.4 Authenticated Group Key Agreement Protocol

2.4.1 Authenticated Group Deffie-Hellman Protocol (A-GDH)

The Authenticated group Deffie-Hellman protocol is proposed by Ateniese et al. (2000). The extended the solution from GDH (Steiner et al., 1998). The implicit key authentication is implemented in the protocol. The member M$_n$ shares a distinct secret K$_{in}$, where $K_{in} = F(\alpha_a^{x_i \cdot x_n} \bmod p)$ with $i \in [1, n-1]$. x_i is long term secret key that selected by every members. The summary protocol is discussed as follows.

Round i

$M_i \rightarrow M_{i+1}$:

$$\{ g^{\frac{s_1 s_2 \cdots s_i}{s_k}} \mid k \in [1, i] \}, g^{s_1 s_2 \cdots s_i}$$

Round n

$M_n \rightarrow M_i$:

$$\{ g^{\frac{s_1 s_2 \cdots s_n}{s_i} K_{in}} \mid i \in [1, n-1] \},$$

for $i \in [1, n-1]$.

Every member computes shared key from $g^{\left(\frac{s_1 s_2 \cdots s_n}{s_i} K_{in}\right) K_{in}^{-1} . s_i} = g^{s_1 s_2 \cdots s_n}$.

2.5 Key agreement based on Braid Group

The computational security of braid groups is based on the difficulty of solving conjugacy and commutator equations in suitably chosen groups. They observe that braid groups are a particularly promising class of groups for the construction of such protocols due to results from Birman, Ko and Lee (1998). This observation was taken up by Ko et al. (2000) who specifies a Diffie-Hellman type key agreement protocol employing commuting one-way functions on braid groups. Some braid group based key agreement protocols were designed in last five years. They gave a specialized version of key agreement protocol based on conjugacy problem. Anshel et al. (2001) put up with their commutator key agreement protocol in the following year. Lee et al. (2006) extend two-party key agreement from Ko et al. (2000) to be the group key agreement on braid group based on the hardness Ko-Lee problem and Cliques. They extended protocol to authenticated group key agreement. Kui et al. (2004) designed the group key agreement based on braid group and Diffie-Hellman key exchanges protocol with dynamic operation protocol including join, leave, merge, partition and refresh protocols.

2.5.1 Preliminaries of Braid Group

The braid groups were first systematically proposed by Emil Artin. He introduced the Artin generators $\sigma_1, \sigma_2, \ldots, \sigma_{n-1}$ for the n strand braid groups what is denoted as B_n. The integer n is called the *braid index* and each element of B_n is called an n-braid. The B_n is a collection of disjoint n strings. A general n-braid is constructed by iteratively applying the σ_i (i = 1,.., n-1) operator, which switches the lower endpoints of the ith and (i+1)th strings keeping the upper endpoints fixed with the (i+1)th string brought above the ith string. If the (i+1)th string passes below the ith string, it is denoted as σ_i^{-1}. Any n-braid can be expressed as a *braid word*, e.g., $\sigma_3\sigma_2\sigma_1^{-1}\sigma_2^{-1}$ is a braid word, a in Figure 2.15, in the braid group B_4. The inverse of braid word is constructed by reversing each crossing sequentially. For example is shown in Figure 2.15, $b = \sigma_1^{-1}\sigma_3^{-1}\sigma_2^{-1}$ and $b^{-1} = \sigma_2\sigma_3\sigma_2$ and The multiplication of two braids word, ab, is the braid achieved by positioning b on the bottom of a. The identity is braid is not intertwining strings.

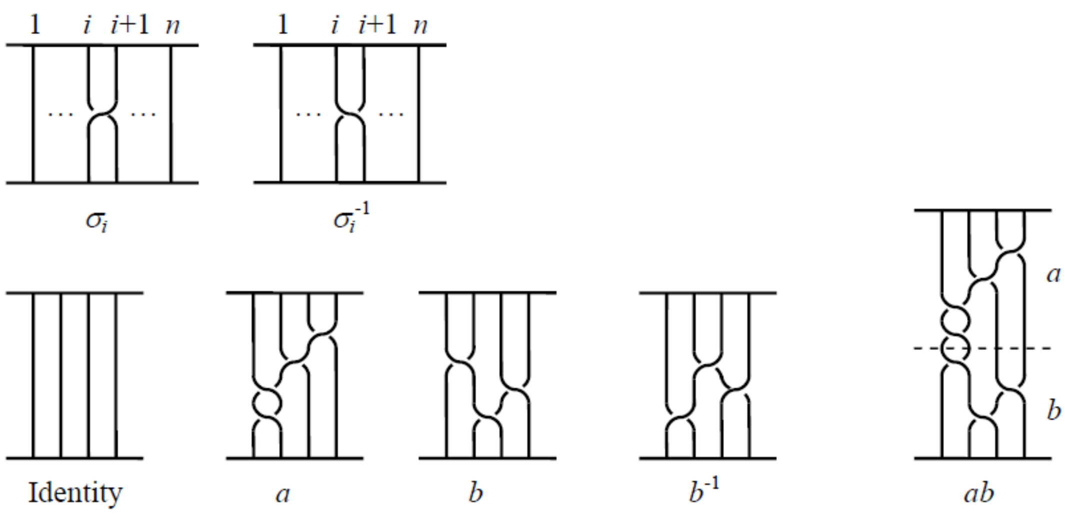


Figure 2.15 Definition of braid groups

The relation of n-braid groups B_n are as follows and shown in Figure 2.16:

1) $\sigma_i \sigma_j = \sigma_j \sigma_i$ where $| i - j | \geq 2$
2) $\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}$

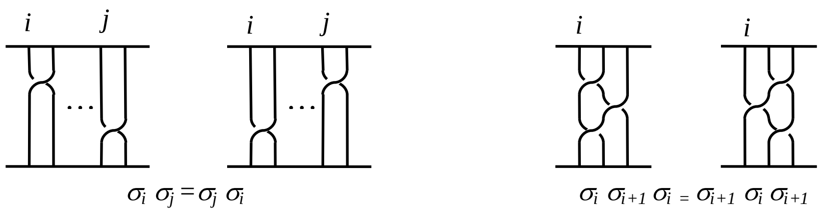

i
j
$\sigma_i \sigma_j = \sigma_j \sigma_i$
$\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}$

Figure 2.16 The relation of braid groups

Product operation and inverse operation can be done in $O(|w|n)$ (Cha, Ko, Lee, Han and Cheon, 2001) where w is the maximum of canonical lengths and n is the braid index. The implementation speed of public-key cryptosystem based on braid groups is much faster than that of ECC and RSA.

2.5.2 Hard Problem in the Braid Groups

The following section explains braid groups in generalized conjugacy search problem (Kim et al., 2004) that is applied to this protocols in order to increasing strength of key. The problem say that x and y are conjugate if there is element a such that $y = a\, x\, a^{-1}$ for $m < n$, B_m can be considered as a subgroup of B_n generated by $\sigma_1, \sigma_2, \ldots, \sigma_{n-1}$.

Instance: $(x, y) \in B_n \times B_n$ such that $y = a\, x\, a^{-1}$ for some $a \in B_n$.

Objective: Find $b \in B_m$ such that $y = b\, x\, b^{-1}$ for $m \leq n$.

Therefore it can conclude that x and y are conjugate. It is easy to compute y when known a and x but the attacks need exponential time to compute b from $b\, x\, b^{-1}$ when known x and y.

They consider two subgroups B_l and B_r of B_{l+r}. The B_l and B_r are made by braiding left l strand and right r strand among $l+r$ strand respectively. The cumulative property for any $a \in B_l$ and $b \in B_r$ is $ab = ba$. The adequately complicated ($l+r$)-braid is selected as $x \in B_{l+r}$. Thus the one-way function is shown as follows:

$$f : B_l \times B_{l+r} \rightarrow B_{l+r} \times B_{l+r}, \quad f(a,x) = (a\, x\, a^{-1}, x)$$

(2.2)

The function is simply to calculate $a\, x\, a^{-1}$ for given a and x but need exponential time to compute a from the information. This one-way function is based on the generalized conjugacy search problem.

2.5.3 Two-party Key Agreement on Braid Groups

There are two types of Two-party key agreement on braid groups. One of them is called Commutator Key Agreement Protocol, presented by Anshel et al (2001), based on combinatorial groups and conjugacy problems. This protocol is not Diffie-Hellman type key agreement protocol.

Ko et al. (2000) proposed a new Diffie-Hellman type key agreement protocol on braid groups based on the hardness of so called Ko-Lee problem. The foundation of this protocol is quite different from widely used protocols on number theory, though there are some similarities in design. This key agreement protocol works as follows:

2.5.3.1 Preparation Step

Suppose A and B want to share a common secret key. An appropriate pair of integers (l, r) and a sufficiently complicated ($l + r$)-braid $\alpha \in B_{l+r}$ is selected and published.

2.5.3.2 Key agreement Scheme

1) A chooses a random secret braid $r_1 \in LB_l$ and sends $y_1 = r_1 \alpha r_1^{-1}$ to B.

2) B chooses a random secret braid $r_2 \in RB_r$ and sends $y_2 = r_2 \alpha r_2^{-1}$ to A.

3) A receives y_2 and computes the shared key $k = r_1 y_2 r_1^{-1}$.

4) B receives y_1 and computes the shared key $k = r_2 y_1 r_2^{-1}$.

Since $r_1 \in LB_l$ and $r_2 \in RB_r$; $r_1 r_2 = r_2 r_1$. This implies $k = r_1 y_2 r_1^{-1} = r_2 y_1 r_2^{-1}$. Therefore *A* and *B* obtain the common secret *k*.

2.5.4 Multiparty Key Agreement on Braid Groupss

Lee et al. (2006) extended above two-party key agreement protocol to the group key agreement protocol by using IKA.1 (GDH.2) structure.

Consider *n* subgroups B_{l1}, B_{l2} , ... , B_{ln} of *l*-braid group B_l where $l = l_1 + l_2 + \ldots + l_n$ for some appropriate integers $l_1, l_2, \ldots, l_n$. Each B_{li} is the subgroup of B_l consisting of braids made by braiding l_i groups from the left among *l*-strands with the order $l_1, l_2, \ldots, l_n$. For any $r_m \in B_{lm}$ and $r_n \in B_{ln}$ with $m \neq n$, $r_m r_n = r_n r_m$. Let $\alpha \in B_l$ be a sufficiently complicated *l*-braid. Supposing $\{M_i \mid i = 1, \ldots, n\}$ is the set of members wishing to share a key. The shared group key is constructed by performing the following steps.

Round i, $i \in [1, n - 1]$:

M_i selects a random $r_i \in B_{li}$, $M_i \rightarrow M_{i+1}$:

$\{ r_i \ldots r^{\wedge}_j \ldots r_1 \alpha r_1^{-1} \ldots r^{\wedge}_j{}^{-1} \ldots r_i^{-1} \mid j = 1,2, \ldots, i \}$ and

$r_i \ldots r_j \ldots r_1 \alpha r_1^{-1} \ldots r_j^{-1} \ldots r_i^{-1}$

where $r^{\wedge}_j$ means that r_j does not exist.

Round n:

M_n selects a random $r_n \in B_{ln}$, $M_n \rightarrow M_i$, $i \in [1, n- 1]$:

$\{ r_n \ldots r^{\wedge}_i \ldots r_1 \alpha r_1^{-1} \ldots r^{\wedge}_i{}^{-1} \ldots r_n^{-1} \}$.

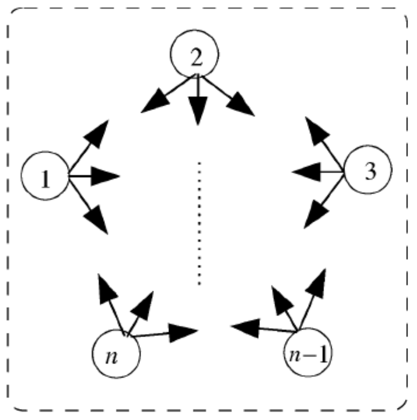
The group key is obtained as $r_n \ldots r_1 \alpha r_1^{-1} \ldots r_n^{-1}$.

2.6 Conceptual Model of Purposed Protocol

2.6.1 Network Assumption

The communication model, the protocol considers group-oriented communication so called many-to-many communication as shown in Figure 2.17; that is, messages are addressed to all the members. For the ease of presentation, in this section, the protocol assumes that all nodes in ad hoc networks are members of a group. In next section, the study discusses how this scheme can be extended for networks where not all nodes are members of a group. For that group-wide symmetric key is used to encrypt group broadcast message. Note that using pairwise shared keys for securing group communication does not improve security in comparison to a scheme based on group keys. This is because under both schemes an adversary only needs to compromise one node to obtain the group data; moreover, if pairwise keys are used for securing group data, a node will have to perform decrypting and re-encrypting for the data packets it is forwarding.

The protocol assume that the resources of a node, such as power, computation and communication capacity, and storage are relatively constrained; thus a node neither can afford public-key operation nor has space for storing pre-deployed pairwise shared keys for all the nodes in the network. However, the protocol assumes that every node has space for storing key tree and computation capacity for computing product and inverse operation.



Senders and Receivers

Figure 2.17 Many-to-many Communication

Once a group is formed, it is ready to be used by other applications. However, in a dynamic group, the views of the underlying group communication system are dynamic. Hence initial group key agreement is only one part. A comprehensive group key agreement scheme must also be able to handle adjustments to group secrets subsequent to all membership change operations in the underlying group communication system. All member operations taking place in this phase can be further classified into two types, addition and exclusion of members. Additive events include addition of single and multiple members, while exclusion events include deletion of single and multiple members.

2.6.2 Network with Non-member Nodes

Multi-hop communications are another possibility in mobile ad hoc networks. Two devices that are mutually unreachable can communicate as long as there is at least one chain of devices that is reachable by both. Multi-hop ad hoc network can be useful consists of several devices, static or dynamic movement, where the devices communication range is extended by using other devices as simple repeaters. It means one or several non-member nodes may be involved in forwarding data packet for group members that are not directly neighboring as shown in Figure 2.18. Although non-member nodes are involved in forwarding the messages for member nodes, they cannot decrypt the message.

2.6.3 Security Assumptions and Attack Models

The security assumptions and attack models is discussed as follows.

2.6.3.1 There is no single member controlling the execution of the protocol but there is some “leader” it is short-lived and restricted to that particular execution of the protocol.

2.6.3.2 The contributory group key management requires each group member to contribute an equal share to the common group key (which is then computed as a function of all members’ contributions). This avoids the problems with the centralized trust and the single point of failure.

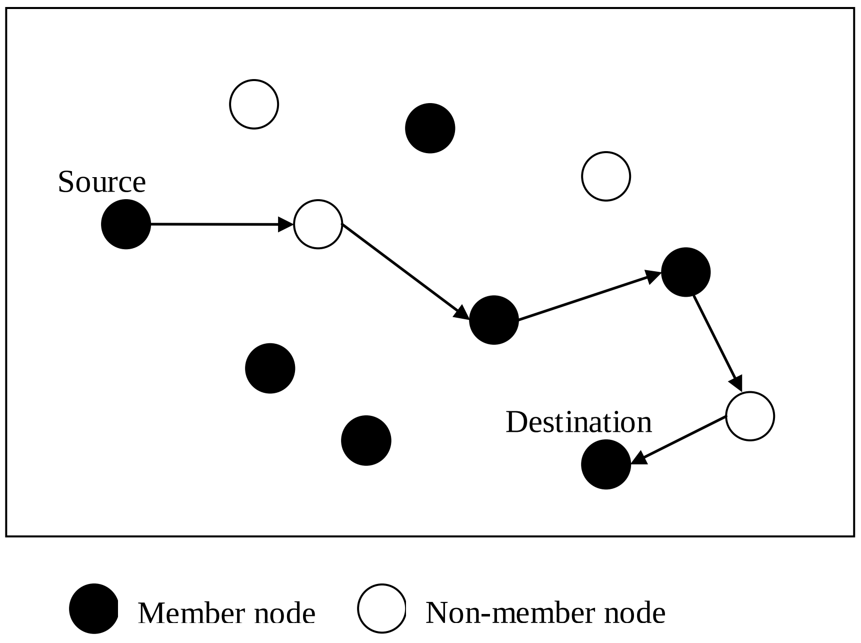


Figure 2.18 Multi-hop Communication with Non-member Nodes

2.6.3.3 If the group memberships are not changed, protocol have to refresh the key periodically, There are two main reasons, one is to limit exposure due to loss of group session keys, the other is to limit the amount of ciphertext available to cryptanalysis for a given group session key. This makes it important for the key refresh protocol not to violate key independence. Additionally, note that the loss of a member's key share can result in the disclosure of all the session keys to which the member has contributed with this share.

2.6.3.4 The solution did not distinguish between an attacker and a compromised node, because the protocol assume that an attacker can obtain all the information stored in a compromised node. Since wireless communication is broadcast-base, assumed that an adversary can eavesdrop on all traffic, inject packets, and replay older packets. Since it is assumed that an adversary can take full control of compromised nodes, an adversary may command compromised nodes to drop off alter messages they are forwarding.

2.6.4 Security Properties

The key derived from the group key agreement protocol needs to meet the following security features:

2.6.4.1 Group key secrecy: It simply means that the derived secret should not be derivable by a non-participant.

2.6.4.2 Forward key secrecy: Merely knowing one of the current group keys, one should not be able to compute previous group keys.

2.6.4.3. Backward key secrecy: Merely knowing one of the current group keys, one should not be able to compute future group keys.

2.6.4.4 Key independence: Knowledge of a subset of group keys in the life-cycle of a group (by a participant or outsider) should not enable knowledge of any key outside this subset.

2.6.4.5 Perfect forward secrecy: Knowledge of a long-term secret key should not enable one to compute past group keys.

2.6.5 Contributory Group Key Agreement

The protocol designed as contributory group key management that requires each group member to contribute an equal share to the common group key (which is then computed as a function of all members' contributions). This avoids the problems with the centralized trust and the single point of failure since mobile ad hoc networks are dynamic topology lead to centralized trust cannot away available.

2.6.6 Tree-based Protocol

To use tree-based contributory group key agreement schemes in this research because is better than other techniques in literature such as CLIQUE. The secret keys are organized in logical tree structure, referred to as the key tree show in Figure 2.19. In the key tree, the root node represents the group key, leaf node represent members' private keys, and each intermediate node corresponds to a subgroup key shared by all the members (leaf node) under this node, The key of each non-leaf node is generated by performing two-party braid group key exchange between the two subgroup represented by its two children.

As a part of protocol, a group member can take on a special sponsor role which involves computing intermediate keys and broadcasting to the group. Each broadcasted message contains the key tree known to source. Any member in the group can unilaterally take on this responsibility, depending on the type of event.

In case of an additive change (join or merge), all group members identify a unique director. This director is responsible for updating its secret key share, computing affected key and blinded key pairs and broadcasting all blinded keys of the new tree to the rest of the group. In response to a subtractive membership change (leave or partition), all members update the tree in the same manner. Group partition results in a smaller tree since some leaf nodes disappear. As a result, some subtrees acquire new siblings; therefore, new intermediate keys and blinded keys must be computed between the new siblings subtrees.

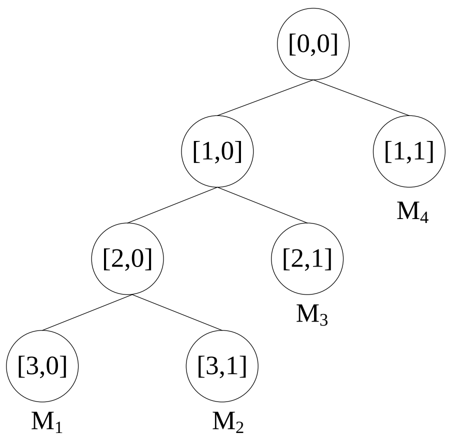


Figure 2.19 Key Tree

2.6.7 Membership Event

A group key agreement will be designed for providing key adjustment protocols stemming from membership changes. The protocol includes in support of the following functions:

Join: a new member is added to the group

Leave: a member is removed from the group

Merge: a group is merged with the current group

Partition: a subset of members is split from the group

Key refresh: the group key is updated

2.6.8 Authentication

In the Diffie-Hellman (DH) scheme, the communication parties at both sides exchange some public information and generate a common session key. Several enhanced DH schemes have been proposed to counter the man-in-the-middle attack. Then authentication is technique to allow the communicating parties to be assured of the others identity for solve man-in-the-middle attack problem. Then, the protocol have long term secret key for authentication purpose and random secret key for generating shared key in group communication.

2.5.9 Braid Group Cryptographic

Many group key agreement protocols have been designed for ad hoc networks. They are all constructed based on generalized Diffie-Hellman key exchange protocol. All these protocols are using modular exponential operation, which itself is inefficiency. Therefore my protocol designed to avoid modular exponential operations since limited computing capability in mobile devices. The braid group based key agreement protocols into ad hoc networks are introduced to mobile ad hoc networks. Moreover, the required computational processes in braid groups techniques are much faster than elliptic curve cryptographic technique because of using just product and inverse operation by avoiding modular exponential operation.

2.6.10 Complexity Analysis

Above the study has discussed the security properties of group key agreement schemes. Important is also their complexity, namely performance costs. Sometimes trade-off between complexity and security is required, so that the schemes are suitable to particular environments. Two of the most important criteria are computation costs and communication costs.

2.6.10.1 Computation Costs

To achieve exact computational costs is impossible and also impracticable. Different implementations of an identical group key agreement scheme bring different results. Even the same implementation cannot guarantee same result in different environments.

However, the protocol can estimate the computation costs by identifying the expensive and time-critical operations. The protocol can ignore the concrete operation time, and only compare the number of such operations. Some computations can be pre-performed before protocol run or computed when the system is idle. Hence only the operations which have to be performed iteratively should be considered.

2.6.10.2 Communication Costs

The communication costs of a group key agreement depend clearly on the topology and properties of the network and the group communication system used. The critical aspects are primarily latency and bandwidth. Additionally the communication costs are implementation-dependent. Hence to achieve fixed communication costs is impracticable. However, the protocol can estimate them by considering the following costs.

1) Number of rounds: this affects serial communication delay. As the number of rounds grows, the communication delay and the probability of message loss or corruption are increased.

2) Total number of messages: as the number of messages grows, the probability of message loss or corruption, and the delay are increased.

3) Number of broadcasts and unicasts: a broadcast operation is much more expensive than a unicast one, since it requires much more acknowledgments with the group communication system. The number of broadcasts should be minimized.

CHAPTER 3

GROUP KEY AGREEMENT USING TREE-BASED BRAID GROUPS

The most of existing group key agreement protocols are based on Diffie-Hellman protocol. The researchers attempted to decrease the number of communication rounds for group members. My protocol is designed based on braid groups cryptographic, Ko-Lee Problem (Ko et al., 2000) in order to reduce computation to linear algebra and based on tree-based group key agreement (Steer, 1988) in order to decrease the number of communication rounds to constant. The protocol is also based on generalized conjugacy search problem that is mentioned in section 2.4.2. Moreover, the protocol is designed with considering the security requirement including group key secrecy, forward secrecy, backward secrecy, and key independent. The protocol is proposed as contributory group key agreement that requires each group member to contribute an equal share to the common group key, i.e. the group key computed as a function of all members' contributions. This avoids the problems with the centralized trust and the single point of failure. The protocol is considered to limit computing, storage and power capacities in ad hoc network. We describe these techniques in following sections. The notations in protocol are denoted as follows:

n	number of protocol participants (group members)
m	number of merging members
i , p, r, d	indices of group members
M_i	i^{th} group member; $i \in \{1, 2, \cdots, n\}$
s_i	session random key of M_i from subgroups (B_{g_i}) of B_g
h	height of tree
T	key tree
T^*	tree after membership operation
$K_{[h, v]}$	secret key at [h, v] node
$BK_{[h, v]}$	blinded key at [h, v] node

$[h, v]$	vth node at level h in a tree

3.1 Key Tree Notation

Key tree is earliest proposed by Wallner, Harder and Agee (1999) and emerged in group key agreement by Kim et al. (2000;2004) in TGDH protocols and Steer et al. (1998) in STR protocol. The tree structure is widely implemented to decrease the communication, computation and storage overhead. The number of communication rounds to form the group key can be reduced to the logarithm of the group size. The braid groups cannot implement with balanced key tree as TGDH protocol, because limitation of braid groups operations and properties. Therefore key tree in this research is based on unbalanced key tree similar to STR protocol. Key tree is implemented in protocol according to be suitable solution for contributory group key agreement in MANET because it does not require that the members are be serialized or structured in order to compute the group. The following section describes the notation and definition of key tree. A sample of key tree based on STR is shown in Figure 3.1. The binary tree, every node is either a leaf or a parent of two nodes, is used in key tree. Each node is represented as $[h,v]$ what is associated with a secret key $K_{[h,\,v]}$ and a blinded key $BK_{[h,\,v]}$. The blinded key is calculated as $f(K_{[h,\,v]})$ where function f ( ) is braid groups key exchange what describe in next section. The members are located at the leaf node. The information of each intermediate node, key and blinded key, is computed from the information of two children nodes to achieve the subgroup key. The leaf node M_i, where $1 \leq i \leq n$, knows every key along the path from node M_i to root node, this path is called the *key-path*. In Figure 3.1, M_1 knows every key { $K_{[3,0]}$, $K_{[2,\,0]\,]}$, $K_{[1,\,0]\ \,]}$, $K_{[0,\,0]}$ } in key-path { [3,0], [2,0], [1,0], [0,0] }. The *co-path* is the set of sibling nodes of each node in the key-path of a member M_i. The sample in Figure 3.1, the co-path of M_1 is set of node { [3,1], [2,1], [1,1] }. The group secret key is key at the root node, $K_{[0,\,0]}$, what can be computed from all blind keys on the co-path and session random $K_{[h,\,v]}$ of a computing node (member).

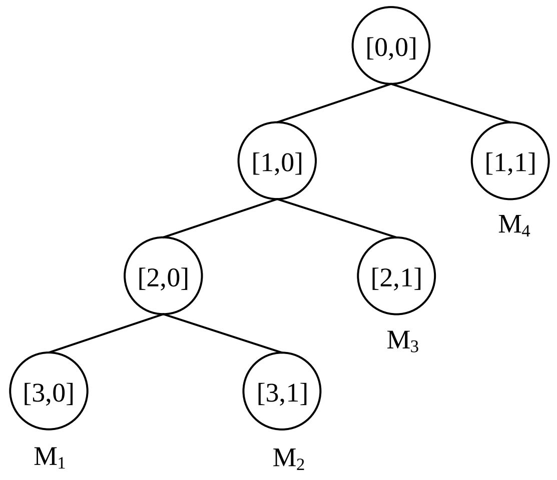


Figure 3.1 Notation of key tree

3.2 Braid Groups Key Exchange

The protocol supposes *n* subgroups (members) $B_{g_1}, B_{g_2}, \ldots, B_{g_n}$ of *g*-braid groups B_g where $g = g_1 + g_2 + \ldots + g_n$. B_g consists of braids made by braiding g_i groups from the left with the order $g_1, g_2, \ldots, g_n$. The braid index of each subgroup, g_i, is not necessarily the same. For any braids $s_l \in B_{gl}$ and $s_m \in B_{gm}$ with $l \neq m$, $s_l s_m = s_m s_l$. The properties of braid groups are applied in this key exchange protocol as follows.

The $\beta_{[h,v]} \in B_q$, where $B_q \subseteq B_g$, be a sufficiently complicated braid are selected and published. Each member selects $\beta_{[h,v]}$ and publishes as public braid word at the leaf node. The $\beta_{[h,v]}$ at intermediate nodes (parent node) including root node is equal to $\beta_{[h+1,2v]}\beta_{[h+1,2v+1]}$. Supposing *n* members need to share a key. Each member selects the secret key from own braid groups. The blinded key $BK_{[h,v]}$ is generated by $f(K_{[h,v]})$ which is equal $K_{[h,v]}\beta_{[h-1,v]}K^{-1}_{[h,v]}$. Therefore key at intermediate nodes $K_{[h,v]}$ are computed as follows:

$$\begin{aligned} K_{[h,v]} &= K_{[h+1,2v]}BK_{[h+1,2v+1]}K^{-1}_{[h+1,2v]} \\ &= K_{[h+1,2v]}K_{[h+1,2v+1]}\beta_{[h,v]}K^{-1}_{[h+1,2v+1]}K^{-1}_{[h+1,2v]} \text{ or} \end{aligned}$$

$$K_{[h,v]} = K_{[h+1,2v+1]}BK_{[h+1,2v]}K^{-1}_{[h+1,2v+1]}$$

$$= K_{[h+1,2v+1]}K_{[h+1,2v]}\beta_{[h,v]}K^{-1}_{[h+1,2v]}K^{-1}_{[h+1,2v+1]}$$

where, for the leaf nodes (members),

$$K_{[h+1,2v]} \in B_{gl} \text{ and } K_{[h+1,2v+1]} \in B_{gm} \text{ with } l \neq m,$$

which have the property

$$K_{[h+1,2v]}\, K_{[h+1,2v+1]} = K_{[h+1,2v+1]}\, K_{[h+1,2v]}.$$

The conclusion of recursive equation is shown as follows:

Base step : for $[h,v]$ which is leaf node,

$$K_{[h,v]} = s_{[h,v]}$$

where $s_{[h,v]}$ is session random key of member at leaf node $[h,v]$

$$BK_{[h,v]} = s_{[h,v]}\beta_{[h-1,0]}s^{-1}_{[h,v]}$$

where $\beta_{[h-1,0]} = \beta_{[h,0]}\beta_{[h,1]}$

Recursive step : for $[h,v]$ which is an intermediate node

$$K_{[h,v]} = K_{[h+1,2v]}BK_{[h+1,2v+1]}K^{-1}_{[h+1,2v]} \text{ or}$$

$$K_{[h,v]} = K_{[h+1,2v+1]}BK_{[h+1,2v]}K^{-1}_{[h+1,2v+1]} \text{ and}$$

$$BK_{[h,v]} = K_{[h,v]}\beta_{[h-1,v]}K^{-1}_{[h,v]}$$

The key generating at $[h, v]$ requires the information composed of key of one child and blinded key of another child. The root key is group secret key that is shared by all current members. A group key can be computed from each member's secret key and all blind keys on the co-path to the root.

An example is shown that all member nodes achieve the same group key in contributory manner. The leaf nodes as A, B and C are labeled for ease to understand as shown in Figure 3.2. Assume each leaf node (member node) select own random secret braid, A select $a \in B_a$, B select $b \in B_b$ and C select $c \in B_c$. The B_a, B_b and B_c are different braid groups, then yield

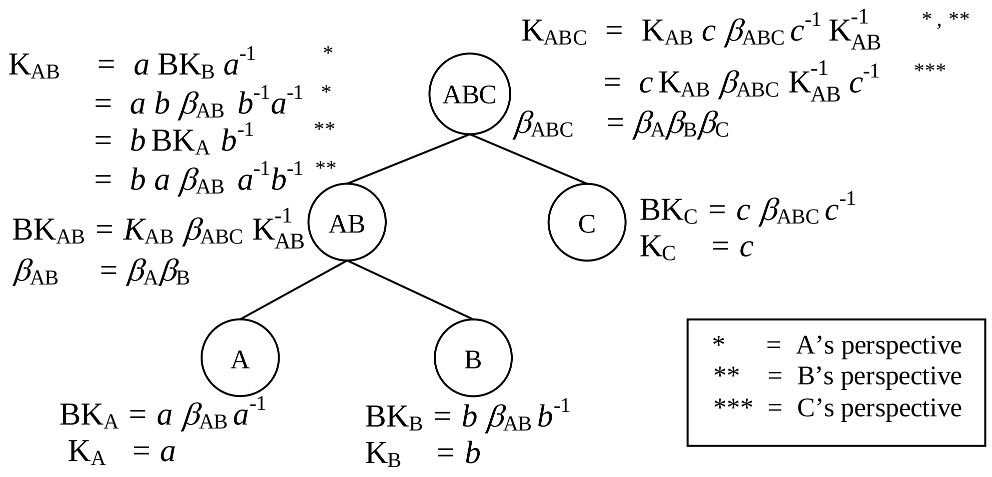
$$ab = ba \text{ and}$$

$$a^{-1}b^{-1} = b^{-1}a^{-1}.$$

Furthermore,

$abc = cba$ and

$a^{-1}b^{-1}c^{-1} = c^{-1}b^{-1}a^{-1}$



$K_{ABC} = K_{AB}\, c\, \beta_{ABC}\, c^{-1} K_{AB}^{-1}$ *, **
$= c\, K_{AB}\, \beta_{ABC}\, K_{AB}^{-1} c^{-1}$ ***
$\beta_{ABC} = \beta_A\beta_B\beta_C$
$K_{AB} = a\, BK_B\, a^{-1}$ *
$= a\, b\, \beta_{AB}\, b^{-1}a^{-1}$ *
$= b\, BK_A\, b^{-1}$ **
$= b\, a\, \beta_{AB}\, a^{-1}b^{-1}$ **
$BK_{AB} = K_{AB}\, \beta_{ABC}\, K_{AB}^{-1}$
$\beta_{AB} = \beta_A\beta_B$
ABC
AB
C
A
B
$BK_C = c\, \beta_{ABC}\, c^{-1}$
$K_C = c$
$BK_A = a\, \beta_{AB}\, a^{-1}$
$K_A = a$
$BK_B = b\, \beta_{AB}\, b^{-1}$
$K_B = b$
* = A's perspective
** = B's perspective
*** = C's perspective

Figure 3.2 The example group key generating

Each member can generate the group key K_{ABC} in contributory manner by recursive equation to achieve as follows:

A's view : $K_{ABC} = a\, b\, \beta_{AB}\, b^{-1}\, a^{-1}\, c\, \beta_{ABC}\, c^{-1}\, a\, b\, \beta_{AB}^{-1}\, b^{-1}\, a^{-1}$

B's view : $K_{ABC} = b\, a\, \beta_{AB}\, a^{-1}\, b^{-1}\, c\, \beta_{ABC}\, c^{-1}\, b\, a\, \beta_{AB}^{-1}\, a^{-1}\, b^{-1}$

C's view : $K_{ABC} = c\, a\, b\, \beta_{AB}\, b^{-1}\, a^{-1}\, \beta_{ABC}\, a\, b\, \beta_{AB}^{-1}\, b^{-1}\, a^{-1}\, c^{-1}$

The braid sequences of root key at each node view, that shown as Figure 3.3 are equal in each subgroup to imply as same braid. The solution that explained above can conclude that braid group can be applied in key tree. Therefore root key that is generated by each member node can be session group key.

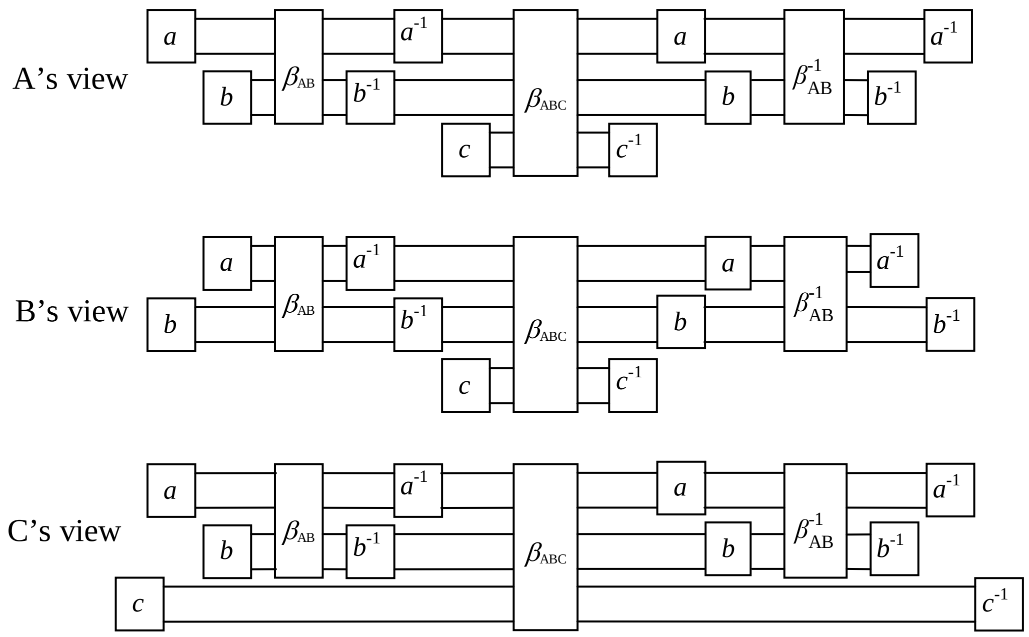


Figure 3.3 Braid Permutation Sequence of each Perspective

3.3 Group Key Agreement on Tree-based Braid Groups (TBG)

The key tree scheme in this research based on STR protocol (Steer et al.,1998) that each node can compute each intermediate key from own secret key and blinded keys of the co-path nodes, therefore the member at leaf node can compute all keys on the key-path. This instance shows that the member needs not to know all blinded keys for generating the group key but knowing the all blinded keys of each member is provided for membership change to be more efficient and robust.

The most of past researches were designed based on position of member in key tree. The scheme may be multi-hop communication between new member and the leader of current group. In other words, some instance new member may contact with the leader that is longest distance comparing with other current members. The communication time between new member and leader is longest. Therefore, in proposed protocols, join and merge event use maximum signal strength for communication in shortest range between new member and leader. The leader in this research is called as **"director"**, therefore the director is assigned momentary dynamic event in order to avoid the single point of failure on existing director. The signal strength achieves from embedded hardware in mobile device such as 802.11b/g. This technique can reduce communication time and transfer information from new member to director as fast as possible.

The following section describes the protocol that constructs the group key management. The protocol includes the following operations:

(1) Join

Join occurs when a potential member wishes to join in an existing group for some reason, such as to share document, to join a conference. The member addition is always performed multi-laterally or, at least, mutually.

(2) Leave

Leave occurs when a member wishes to leave the group, or is forced to leave the group. In the former case, member deletion is mutual, while in the later case unilateral. There may be various reasons for a member to be forced to leave a group, such as involuntary disconnect or forced exclusion. A group key agreement scheme does not need to bother about reasons. It leaves the underlying group communication system to handle it. However, it must adjust the group key on this change.

(3) Merge

Another existing group wishes to merge into the current group to form a super-group. The merge of more than two groups can be considered to be subsequent merging of two groups. All members have the same view after group merge is handled. Correspondingly, a group merge can be either voluntary or involuntary with reasons including the network fault heal and explicit merge. The network fault heal occurs when a network event causes previously disconnected network partitions reconnect. Consequently, all groups formed after partition are merged into a single group. Otherwise, the explicit (application-driven) merge occurs when the application decides to merge multiple pre-existing groups into a single group.

Due to the properties of ad hoc networks, network failures and network fault heals are both common and expected. Hence dealing with group partitions and merges is a crucial component of group key agreement for ad hoc networks.

(4) Partition

Partition occurs when a subset of members request to split from the current group. Group partition can be considered of having multiple leaves, subgroup leave and combination between multiple (or single) leaves and to form sub-groups of remaining members. First, the multiple leaves occurs when multiple members leave the group without forming own subgroup. Second, the subgroup leave occurs when all members leave the group and form own subgroup. After group partition, member's group view is relative to the subgroup it belongs to. For example, there is a group G consisting of six members M_1, M_2, M_3, M_4, M_5, and M_6. Now the group G is split into two smaller groups: group G_1 with members M_1, M_3, M_5, and group G_2 with members M_2, M_4, and M_6. All members in G_1 see M_2, M_4, and M_6 as leaving members, while all members in G_2 see M_1, M_3, and M_5 as leaving members. Third, the partition occurs when some member leaves the group causes split the remaining members into form sub-groups. For example similar to second scheme, M_3 and M_4 leave from the group, and then the remaining members are M_1, M_2, M_5, and M_6. The M_3 and M_4 leaving from the group cause the remaining members split into two groups: group G_1 with members M_1 and M_5, and group G_2 with members M_2, and M_6. A group partition can take place for several reasons such as network failure and explicit partition. The network failure occurs when a network event causes dis-connectivity within the group. Consequently, a group is split into fragments some of which are singletons while others (those that maintain mutual connectivity) are sub-groups. Otherwise, the explicit (application-driven) partition occurs when the application decides to split the group into multiple components or simply exclude multiple members at once.

(5) Key Refresh

It is desirable to refresh the key periodically, even if the group memberships are not changed. There are two main reasons, one is to limit exposure due to loss of group session keys, the other is to limit the amount of ciphertext available to cryptanalysis for a given group session key. This makes it important for the key refresh protocol not to violate key independence. Additionally, note that the loss of a member's key share can result in the disclosure of all the session keys to which the member has contributed with this share. Therefore, not only session keys, but also the individual key shares must be refreshed periodically.

The following section describes the protocol that constructs the group key agreement.

3.3.1 Setup Protocol

The members who want to form a group can be ordered according to some criteria such as MAC address of device. The structure of the key tree can be then derived from this order. The first member in the order is selected as director. The blinded key of member M_i is $BK_i = s_i \beta_r \beta_i s_i^{-1}$ where β_r is existing publish braid word at root node before the director will update next member to key tree by order. Each member knows the own β_r because it have some criteria such as MAC address of all members. It can order the MAC address by itself, and then it knows sequence of members. After the director received blinded key from each member sends, it create key tree and computes keys and blinded keys of intermediate node in key tree. Later the director broadcasts the key tree to all members in the group. The process is illustrated as follows:

Step 1: Each M_i, $i \in \{1, \cdot\cdot\cdot, n\}$ sends its blinded session random key to director M_d.

$$\{ M_i , i \in [1, n] \} - M_d \xrightarrow{BK_i} M_d$$

Step 2: The director computes recursively keys and blinded keys to the root and broadcasts the key tree containing the all blinded key.

$$M_d \xrightarrow{T[BK]} \{ M_i , i \in [1, n] \} - M_d$$

Step 3: Each member computes the secret group key.

Then total communication message in setup protocol is n rounds including the setup message from each member to director and key tree information from director to all members.

3.3.2 Join Protocol

The group has n members, $\{M_1,..,M_n\}$. Every member in current group knows the existing key tree. The new member M_{n+1} wishes to join the group by detecting the maximum signal strength of current group member to be as director in order to communicate in one hop and shortest distance. Later the new member sends, JOIN_MESSAGE, request message to director. The director refreshes the own session random key, computes keys and blinded keys of intermediate nodes up to the root node, and sends the existing key tree with its new session random key to new member. The insertion point of new member on key tree will be new root node of key tree because the new member can computed the information of new key tree with the lowest computation cost. The new member needs to compute only the blinded key at the new root node. Later, the new member updates existing key tree in accordance with creates a new root node and a new member node. Next, the new member selects session random key (i.e., secret key) and computes keys and blinded keys going up to the root. The blinded key of new member M_{n+1} is BK_{n+1} = $s_{n+1}\beta_r\beta_{n+1}s_{n+1}^{-1}$ where β_r is existing publish braid word at root node that the new member can find in existing key tree information. The new member broadcasts the new key tree containing only blinded keys to all other members. Finally all other members compute the new group key. This join protocol provides backward secrecy since director updated session random key that knowledge of a new member is unable to compute old group keys. Figure 3.4 shows situation before new member joins. After that, the Figure 3.5 shows an example of M_4 joining a group where director as M_2. This instance, it means that the M_2 is nearest with M_4. The conclusion of join protocol is illustrated as follows:

Step 1: The new member detects the maximum signal strength of current group members as director and sends JOIN_MESSAGE request message to join the group. After the director received the request message, it selected its new session random key, computes keys and blinded keys, and sends the existing key tree to new member.

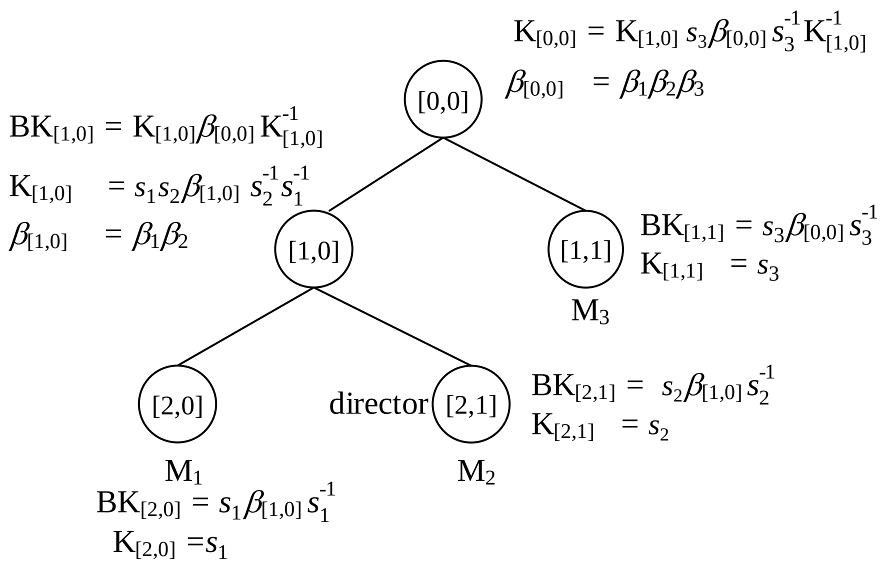
$$M_d \xrightarrow{T[BK]} M_{n+1}$$

Step 2: The new member selected its session random key, updates key tree, computes keys and blinded keys, and broadcasts the new key tree containing the only all blinded key.

$$M_{n+1} \xrightarrow{T^*[BK]} \{ M_i , i \in [1, n] \}$$

Step 3: Each member computes the secret group key.

Then total communication message in join protocol is two rounds including existing key tree information from director to new member and new key tree information from new member to all members. There are *n* serial numbers of braid permutation in the worst case if director is deepest node.



$K_{[0,0]} = K_{[1,0]} s_3 \beta_{[0,0]} s_3^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_1 \beta_2 \beta_3$
[0,0]
$BK_{[1,0]} = K_{[1,0]} \beta_{[0,0]} K_{[1,0]}^{-1}$
$K_{[1,0]} = s_1 s_2 \beta_{[1,0]} s_2^{-1} s_1^{-1}$
$\beta_{[1,0]} = \beta_1 \beta_2$
[1,0]
[1,1]
$BK_{[1,1]} = s_3 \beta_{[0,0]} s_3^{-1}$
$K_{[1,1]} = s_3$
M_3
[2,0]
director
[2,1]
$BK_{[2,1]} = s_2 \beta_{[1,0]} s_2^{-1}$
$K_{[2,1]} = s_2$
M_1
M_2
$BK_{[2,0]} = s_1 \beta_{[1,0]} s_1^{-1}$
$K_{[2,0]} = s_1$

Figure 3.4 Before tree updated: M_4 join, M_2 as director

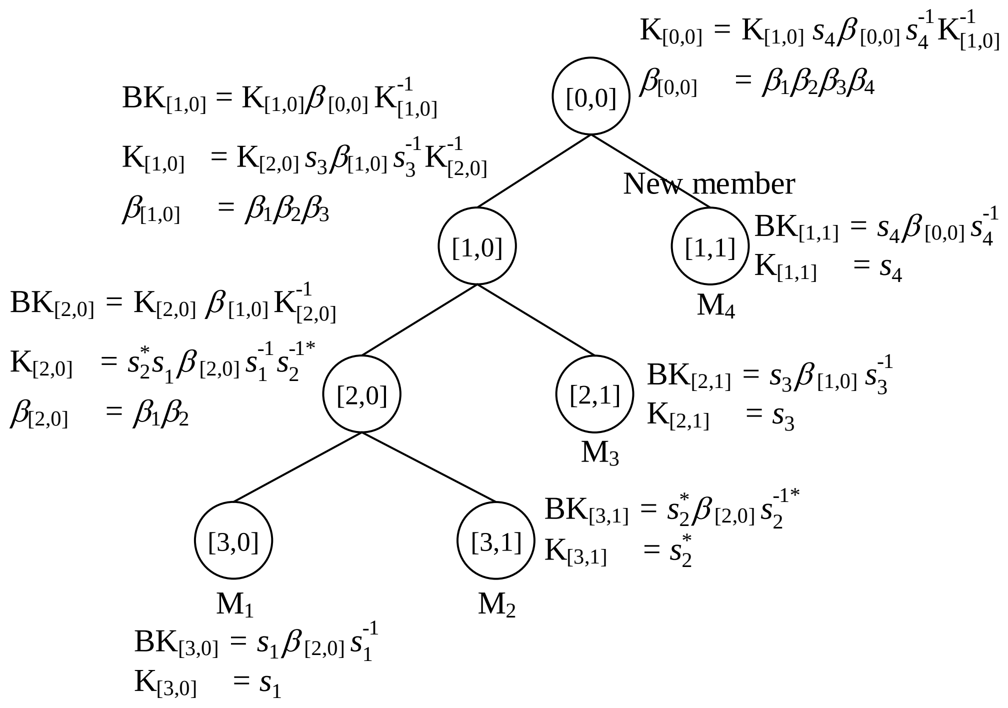

$K_{[0,0]} = K_{[1,0]}\, s_4 \beta_{[0,0]}\, s_4^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_1\beta_2\beta_3\beta_4$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\beta_{[0,0]}\, K_{[1,0]}^{-1}$
$K_{[1,0]} = K_{[2,0]}\, s_3\beta_{[1,0]}\, s_3^{-1} K_{[2,0]}^{-1}$
$\beta_{[1,0]} = \beta_1\beta_2\beta_3$
New member
[1,0]
[1,1]
$BK_{[1,1]} = s_4\beta_{[0,0]}\, s_4^{-1}$
$K_{[1,1]} = s_4$
M_4
$BK_{[2,0]} = K_{[2,0]}\, \beta_{[1,0]}\, K_{[2,0]}^{-1}$
$K_{[2,0]} = s_2^* s_1 \beta_{[2,0]}\, s_1^{-1} s_2^{-1*}$
$\beta_{[2,0]} = \beta_1\beta_2$
[2,0]
[2,1]
$BK_{[2,1]} = s_3\beta_{[1,0]}\, s_3^{-1}$
$K_{[2,1]} = s_3$
M_3
$BK_{[3,1]} = s_2^*\beta_{[2,0]}\, s_2^{-1*}$
$K_{[3,1]} = s_2^*$
[3,0]
[3,1]
M_1
M_2
$BK_{[3,0]} = s_1\beta_{[2,0]}\, s_1^{-1}$
$K_{[3,0]} = s_1$

Figure 3.5 After tree updated: M_4 join, M_2 as director

3.3.3 Leave Protocol

The protocol begins with *n* current members and the member M_r wants to leave the group. In this event the director is the leaf node above the removing node in existing key tree before leave event. In special case, if the leaving node is child of the root, the director is leaf node below the removing node. Because director only calculates the new blinded key of intermediated nodes above director up to the root node, other intermediated nodes is not necessary to update blinded keys. Upon hearing the leave event from the group, the director updates key tree by deleting the leaf node of M_r, selects a new secret session random key and computes keys and blinded keys going up to the root. The director computes the new blinded key of leaf nodes below removing node to achieve as $BK_{pre} = BK_{old}\, \beta_l^{-1}\beta_d$. Also the director computes the new blinded key of leaf nodes above director to yield as $BK_{pre} = BK_{old}\, \beta_l^{-1}$. In the formula, β_l is public braid of removing node and β_d is public braid of director. Next, the director broadcasts the new key tree containing only blinded keys to all other members. Then the remainder members compute the new group key. Figure 3.6 shows situation before a member leaves. After that Figure 3.7 shows the example of M_2 leaving a group where director as M_3. The conclusion of leave protocol is illustrated as follows:

Step 1: The director selects the new session random key, updates the key tree, computes keys and blinded keys and broadcasts the new key tree.

$$M_d \xrightarrow{T^*[BK]} \{ M_i , i \in [1, n] \} - M_r$$

Step 2: Each member computes the secret group key.

Then total communication message in leave protocol is one round. In the worst case, the serial number of braid permutation in this protocol is equal to n -1 when the leaving node is deepest leaf.

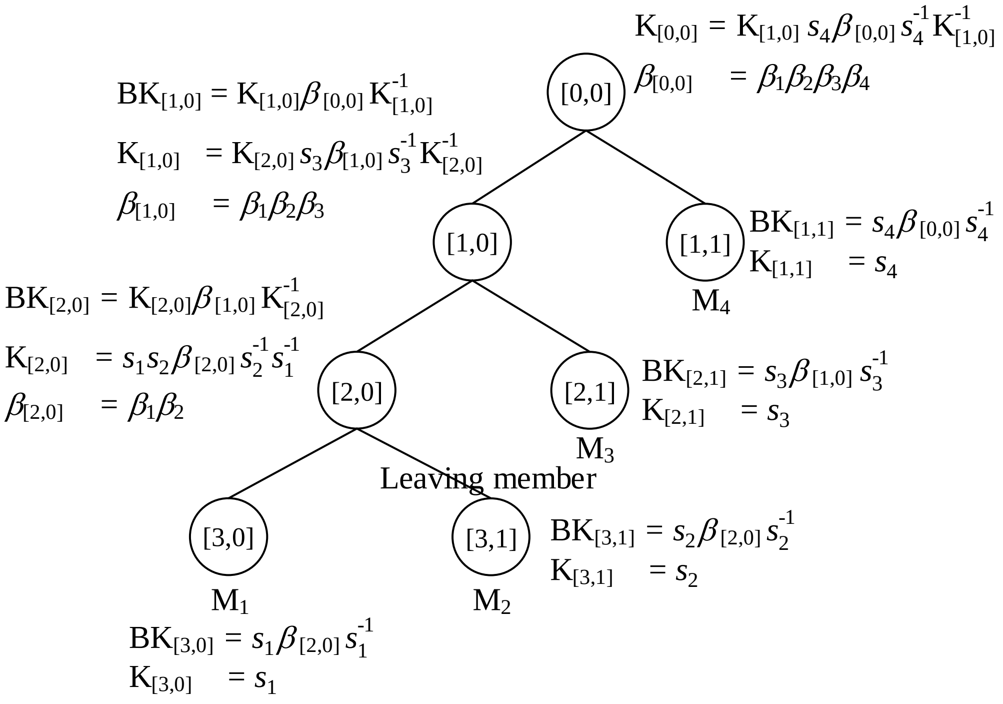

$K_{[0,0]} = K_{[1,0]} s_4 \beta_{[0,0]} s_4^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_1\beta_2\beta_3\beta_4$
[0,0]
$BK_{[1,0]} = K_{[1,0]} \beta_{[0,0]} K_{[1,0]}^{-1}$
$K_{[1,0]} = K_{[2,0]} s_3 \beta_{[1,0]} s_3^{-1} K_{[2,0]}^{-1}$
$\beta_{[1,0]} = \beta_1\beta_2\beta_3$
[1,0]
[1,1]
$BK_{[1,1]} = s_4 \beta_{[0,0]} s_4^{-1}$
$K_{[1,1]} = s_4$
M_4
$BK_{[2,0]} = K_{[2,0]} \beta_{[1,0]} K_{[2,0]}^{-1}$
$K_{[2,0]} = s_1 s_2 \beta_{[2,0]} s_2^{-1} s_1^{-1}$
$\beta_{[2,0]} = \beta_1\beta_2$
[2,0]
[2,1]
$BK_{[2,1]} = s_3 \beta_{[1,0]} s_3^{-1}$
$K_{[2,1]} = s_3$
M_3
Leaving member
[3,0]
[3,1]
$BK_{[3,1]} = s_2 \beta_{[2,0]} s_2^{-1}$
$K_{[3,1]} = s_2$
M_1
M_2
$BK_{[3,0]} = s_1 \beta_{[2,0]} s_1^{-1}$
$K_{[3,0]} = s_1$

Figure 3.6 Before tree updated: M_2 leave, M_3 as director

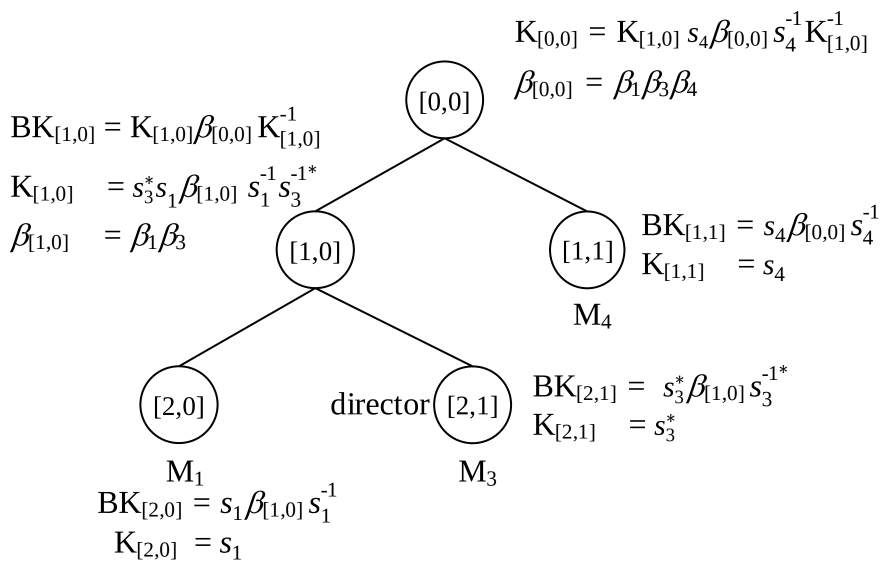

$K_{[0,0]} = K_{[1,0]}\, s_4 \beta_{[0,0]}\, s_4^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_1\beta_3\beta_4$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\beta_{[0,0]}\, K_{[1,0]}^{-1}$
$K_{[1,0]} = s_3^* s_1 \beta_{[1,0]}\, s_1^{-1} s_3^{-1*}$
$\beta_{[1,0]} = \beta_1\beta_3$
[1,0]
[1,1]
$BK_{[1,1]} = s_4\beta_{[0,0]}\, s_4^{-1}$
$K_{[1,1]} = s_4$
M_4
[2,0]
director
[2,1]
$BK_{[2,1]} = s_3^*\beta_{[1,0]}\, s_3^{-1*}$
$K_{[2,1]} = s_3^*$
M_1
M_3
$BK_{[2,0]} = s_1\beta_{[1,0]}\, s_1^{-1}$
$K_{[2,0]} = s_1$

Figure 3.7 After tree updated: M_2 leave, M_3 as director

3.3.4 Merge Protocol

In this instant assume that *m* merging group needs to merge with *c* current group. The existing merging group director detects to achieve maximum signal strength what is measured as the closest member between itself and current group members. The current group director is the member what has maximum signal strength with merging group director. After the merging process, the leftest leaf of shorter tree becomes the right child of a new intermediate node. The root of the longer tree is left child of the new intermediate node.

After the current group director received the MERGE_MESSAGE message, it refreshes session random key, computes keys and blinded keys, and sends the current group's key tree containing the all blinded keys to merging group director. Later, the merging group director updates key tree by combining the merging group's key tree and current group's key tree at the new root node, the director chooses session random key, computes keys and blinded keys up to the root node, and broadcasts new key tree containing the all blinded keys to all members in new group. Finally, the group key is calculated independently by each member. Figure 3.8 shows the initial situation before current group merges with other group. After that Figure 3.9 shows the example of merge operation. The member that has maximum signal strength of merging group director, M_5, is M_1, and then M_1 is current group director. The conclusion of merge protocol is shown as follows:

Step 1: The director of current group what selects new session random key, computes blinded keys and sends updated key tree to the merging group director.

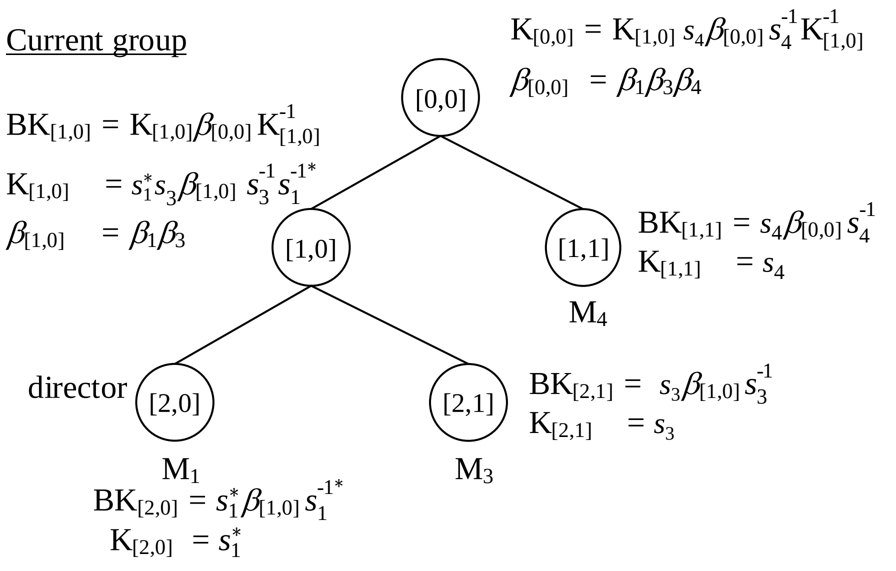
$$M_{d[c]} \xrightarrow{T_c^*[BK]} M_{d[m]}$$

Step 2: The merging group director selects its new session random key, combines key tree, computes the all blinded keys, and broadcasts the new key tree containing the only all blinded keys.

$$M_{d[m]} \xrightarrow{T^*[BK]} \{ M_i \text{, } i \in [1, n+m] \}$$

Step 3: Each member computes the secret group key.

Then total communication message in merge protocol is two rounds. The serial number of braid permutation in merge protocol is equal to $n+m$ where m is amount of merging group member.


Current group
$K_{[0,0]} = K_{[1,0]}\, s_4 \beta_{[0,0]}\, s_4^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_1\beta_3\beta_4$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\beta_{[0,0]}\, K_{[1,0]}^{-1}$
$K_{[1,0]} = s_1^* s_3 \beta_{[1,0]}\, s_3^{-1} s_1^{-1*}$
$\beta_{[1,0]} = \beta_1\beta_3$
[1,0]
[1,1]
$BK_{[1,1]} = s_4\beta_{[0,0]}\, s_4^{-1}$
$K_{[1,1]} = s_4$
M_4
director
[2,0]
[2,1]
$BK_{[2,1]} = s_3\beta_{[1,0]}\, s_3^{-1}$
$K_{[2,1]} = s_3$
M_1
M_3
$BK_{[2,0]} = s_1^*\beta_{[1,0]}\, s_1^{-1*}$
$K_{[2,0]} = s_1^*$

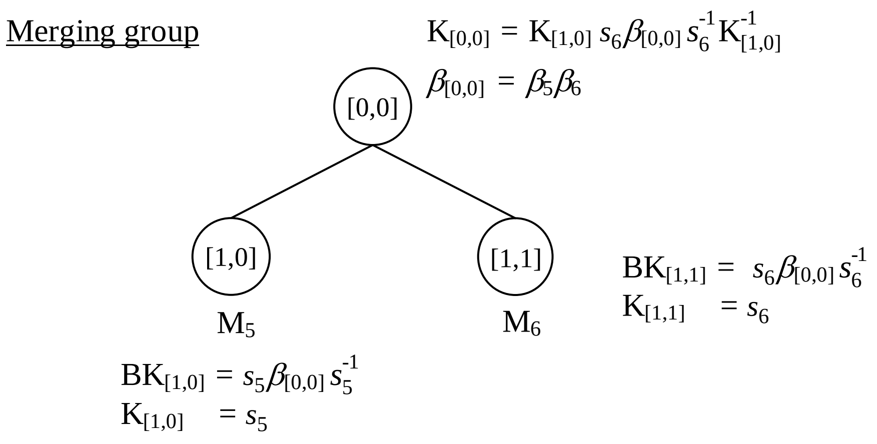
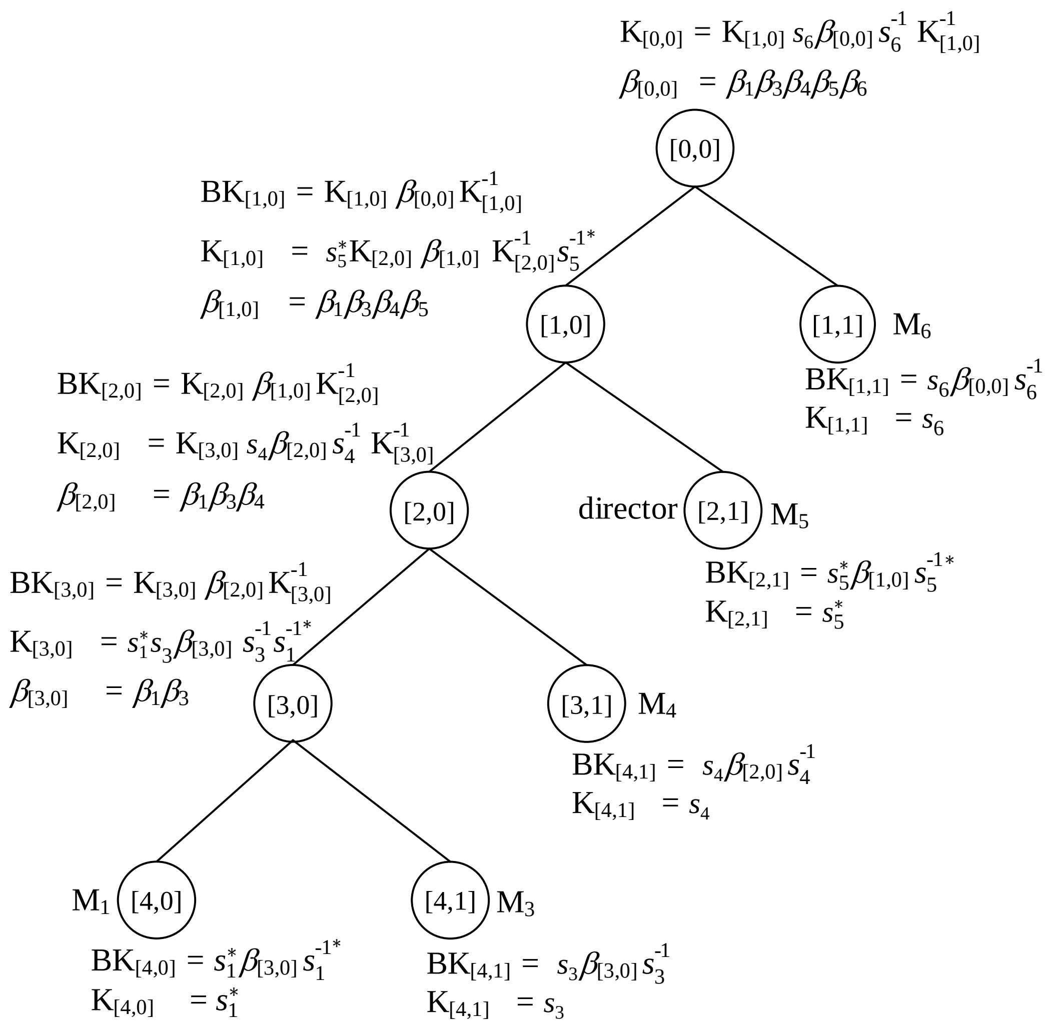

Merging group
$K_{[0,0]} = K_{[1,0]}\, s_6\beta_{[0,0]}\, s_6^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_5\beta_6$
[0,0]
[1,0]
[1,1]
$BK_{[1,1]} = s_6\beta_{[0,0]}\, s_6^{-1}$
$K_{[1,1]} = s_6$
M_5
M_6
$BK_{[1,0]} = s_5\beta_{[0,0]}\, s_5^{-1}$
$K_{[1,0]} = s_5$

Figure 3.8 Before tree updated: Merge Protocol

3.3.5 Partition Protocol

The partition operation occur many reasons that explained in section 3.3 (1). When multiple members *p* need to leave the group, the director is the node above the undermost removing nodes in existing key tree. Otherwise, if the leaving node is child of root and the undermost removing nodes does not exist, the director is leaf node below the undermost the removing nodes. Because director only calculates the new blinded key of intermediate nodes above director up to the root node, other intermediate nodes are not necessary to update blinded keys. It means that amount of new blinded keys calculation is least. Moreover, the director computes the new blinded keys of leaf nodes below and above the director in the same manner as leave protocol. There are three examples in partition protocol. First, the partition protocol actually presents a concurrent multiple members, *p*, leaving from group. As for the leave protocol, after the director deletes all leaving members from key tree, it selects new session random key, computes keys and blinded keys going up to the root, and broadcasts the key tree with blinded keys to reminder members. Finally, each member computes the new group key. Second, the group needs to split into sub-group.



$K_{[0,0]} = K_{[1,0]} s_6 \beta_{[0,0]} s_6^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_1\beta_3\beta_4\beta_5\beta_6$
[0,0]
$BK_{[1,0]} = K_{[1,0]} \beta_{[0,0]} K_{[1,0]}^{-1}$
$K_{[1,0]} = s_5^* K_{[2,0]} \beta_{[1,0]} K_{[2,0]}^{-1} s_5^{-1*}$
$\beta_{[1,0]} = \beta_1\beta_3\beta_4\beta_5$
[1,0]
[1,1]
M_6
$BK_{[1,1]} = s_6 \beta_{[0,0]} s_6^{-1}$
$K_{[1,1]} = s_6$
$BK_{[2,0]} = K_{[2,0]} \beta_{[1,0]} K_{[2,0]}^{-1}$
$K_{[2,0]} = K_{[3,0]} s_4 \beta_{[2,0]} s_4^{-1} K_{[3,0]}^{-1}$
$\beta_{[2,0]} = \beta_1\beta_3\beta_4$
[2,0]
director
[2,1]
M_5
$BK_{[2,1]} = s_5^* \beta_{[1,0]} s_5^{-1*}$
$K_{[2,1]} = s_5^*$
$BK_{[3,0]} = K_{[3,0]} \beta_{[2,0]} K_{[3,0]}^{-1}$
$K_{[3,0]} = s_1^* s_3 \beta_{[3,0]} s_3^{-1} s_1^{-1*}$
$\beta_{[3,0]} = \beta_1\beta_3$
[3,0]
[3,1]
M_4
$BK_{[4,1]} = s_4 \beta_{[2,0]} s_4^{-1}$
$K_{[4,1]} = s_4$
M_1
[4,0]
[4,1]
M_3
$BK_{[4,0]} = s_1^* \beta_{[3,0]} s_1^{-1*}$
$K_{[4,0]} = s_1^*$
$BK_{[4,1]} = s_3 \beta_{[3,0]} s_3^{-1}$
$K_{[4,1]} = s_3$

Figure 3.9 After tree updated: Merge Protocol

As first scheme, after director of subgroups deletes leaving members of other groups, each director selects new own session random key, computes keys and blinded keys going up to the root, and broadcasts the key tree with blinded keys to sub-group members. Finally, each member computes the new group key. Third, leaving member(s) causes to spilt remaining members to subgroup. After leaving member(s) leave, remaining members form the subgroup what both sub-group members see other sub-group members and leaving member(s) as all leaving members. Figure 3.10 shows the initial situation before members leave the group. First scheme, multiple members leave from the group. Figure 3.11 show the example when M_4 and M_5 as leaving members and the director as M_6. Second scheme the members are divided into two smaller groups. Figure 3.12 shows an example of partition operation of second scheme when all members in G_2 see M_1, M_3, and M_6 as leaving members, while all members in G_1 see M_4 and M_5 as leaving members. The director of G_1 is M_6 and of G_2 is M_4. Third scheme, leaving member causes the group partition to two sub-groups. Figure 3.13 shows third scheme when all members in G_1 see M_4, M_5, and M_6 as leaving members, while all members in G_2 see M_1, M_3 and M_4 as leaving members. The director of G_1 is M_3 and of G_2 is M_5. The conclusion of partition protocol is shown as follows:

Step 1: The director updates the key tree, selected the new session random key, computes keys and blinded keys and broadcasts the new key tree to remaining members.

$$M_d \xrightarrow{T^*[BK]} \{\, M_i\,,\ i \in [1, n\text{-}p]\ \}$$

Step 2: Each member computes the secret group key.

Then total communication message in partition protocol is one round. The serial number of braid permutation in partition protocol is equal to n - p where p is amount of partition member.

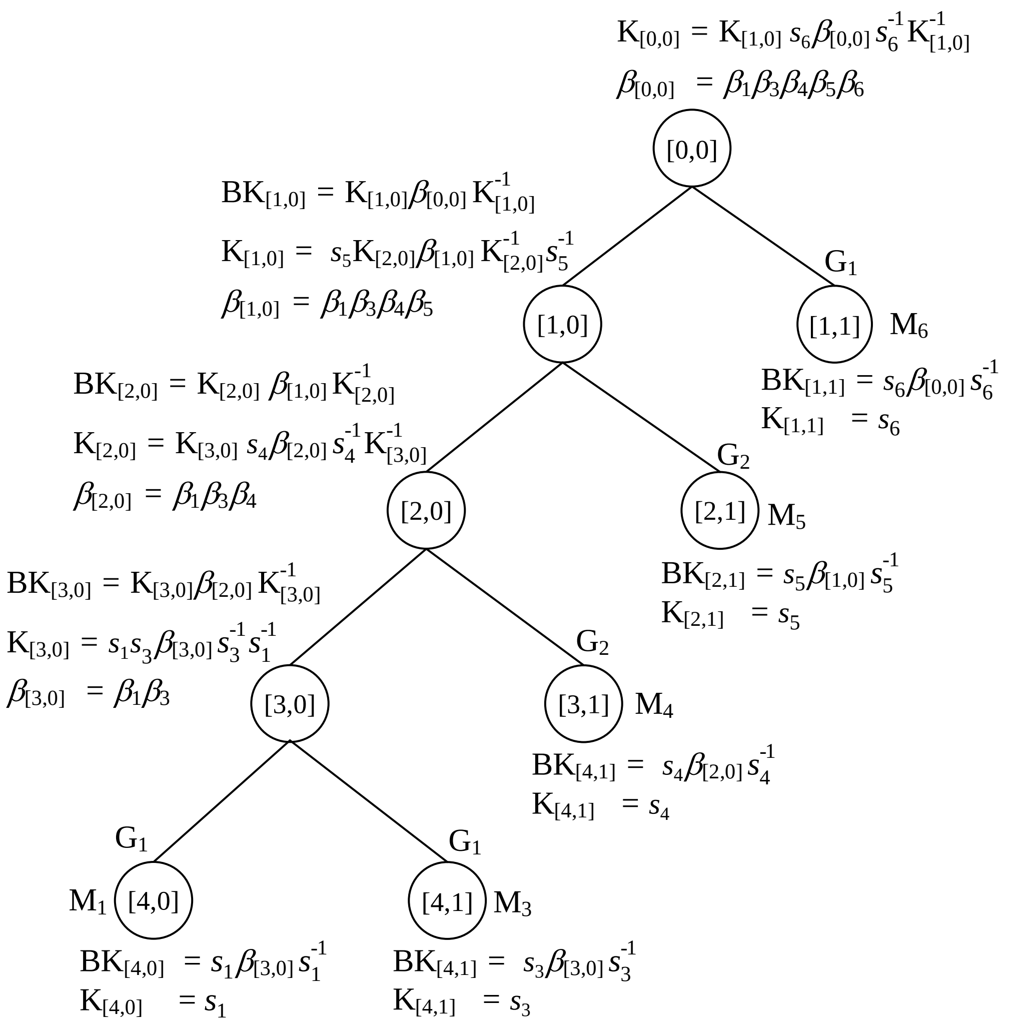

$K_{[0,0]} = K_{[1,0]}\, s_6 \beta_{[0,0]}\, s_6^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_1\beta_3\beta_4\beta_5\beta_6$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\beta_{[0,0]} K_{[1,0]}^{-1}$
$K_{[1,0]} = s_5 K_{[2,0]}\beta_{[1,0]} K_{[2,0]}^{-1} s_5^{-1}$
$\beta_{[1,0]} = \beta_1\beta_3\beta_4\beta_5$
G_1
[1,0]
[1,1]
M_6
$BK_{[1,1]} = s_6\beta_{[0,0]} s_6^{-1}$
$K_{[1,1]} = s_6$
$BK_{[2,0]} = K_{[2,0]}\, \beta_{[1,0]} K_{[2,0]}^{-1}$
$K_{[2,0]} = K_{[3,0]}\, s_4\beta_{[2,0]} s_4^{-1} K_{[3,0]}^{-1}$
$\beta_{[2,0]} = \beta_1\beta_3\beta_4$
G_2
[2,0]
[2,1]
M_5
$BK_{[2,1]} = s_5\beta_{[1,0]} s_5^{-1}$
$K_{[2,1]} = s_5$
$BK_{[3,0]} = K_{[3,0]}\beta_{[2,0]} K_{[3,0]}^{-1}$
$K_{[3,0]} = s_1 s_3\beta_{[3,0]} s_3^{-1} s_1^{-1}$
$\beta_{[3,0]} = \beta_1\beta_3$
G_2
[3,0]
[3,1]
M_4
$BK_{[4,1]} = s_4\beta_{[2,0]} s_4^{-1}$
$K_{[4,1]} = s_4$
G_1
G_1
M_1
[4,0]
[4,1]
M_3
$BK_{[4,0]} = s_1\beta_{[3,0]} s_1^{-1}$
$K_{[4,0]} = s_1$
$BK_{[4,1]} = s_3\beta_{[3,0]} s_3^{-1}$
$K_{[4,1]} = s_3$

Figure 3.10 Before tree updated: Partition Protocol

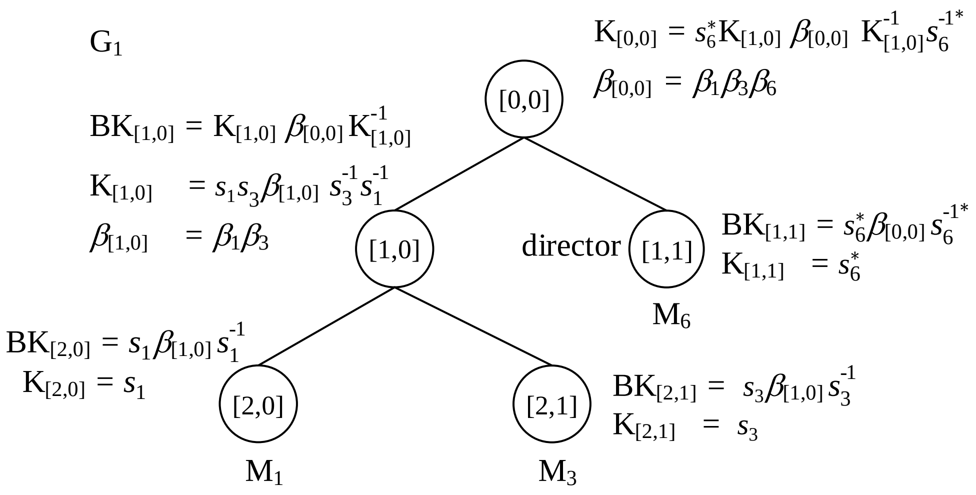

G_1
$K_{[0,0]} = s_6^{*} K_{[1,0]}\, \beta_{[0,0]}\, K_{[1,0]}^{-1} s_6^{-1*}$
$\beta_{[0,0]} = \beta_1\beta_3\beta_6$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\, \beta_{[0,0]} K_{[1,0]}^{-1}$
$K_{[1,0]} = s_1 s_3\beta_{[1,0]}\, s_3^{-1} s_1^{-1}$
$\beta_{[1,0]} = \beta_1\beta_3$
[1,0]
director
[1,1]
M_6
$BK_{[1,1]} = s_6^{*}\beta_{[0,0]} s_6^{-1*}$
$K_{[1,1]} = s_6^{*}$
$BK_{[2,0]} = s_1\beta_{[1,0]} s_1^{-1}$
$K_{[2,0]} = s_1$
[2,0]
[2,1]
$BK_{[2,1]} = s_3\beta_{[1,0]} s_3^{-1}$
$K_{[2,1]} = s_3$
M_1
M_3

Figure 3.11 After tree updated: Partition Protocol in first scheme

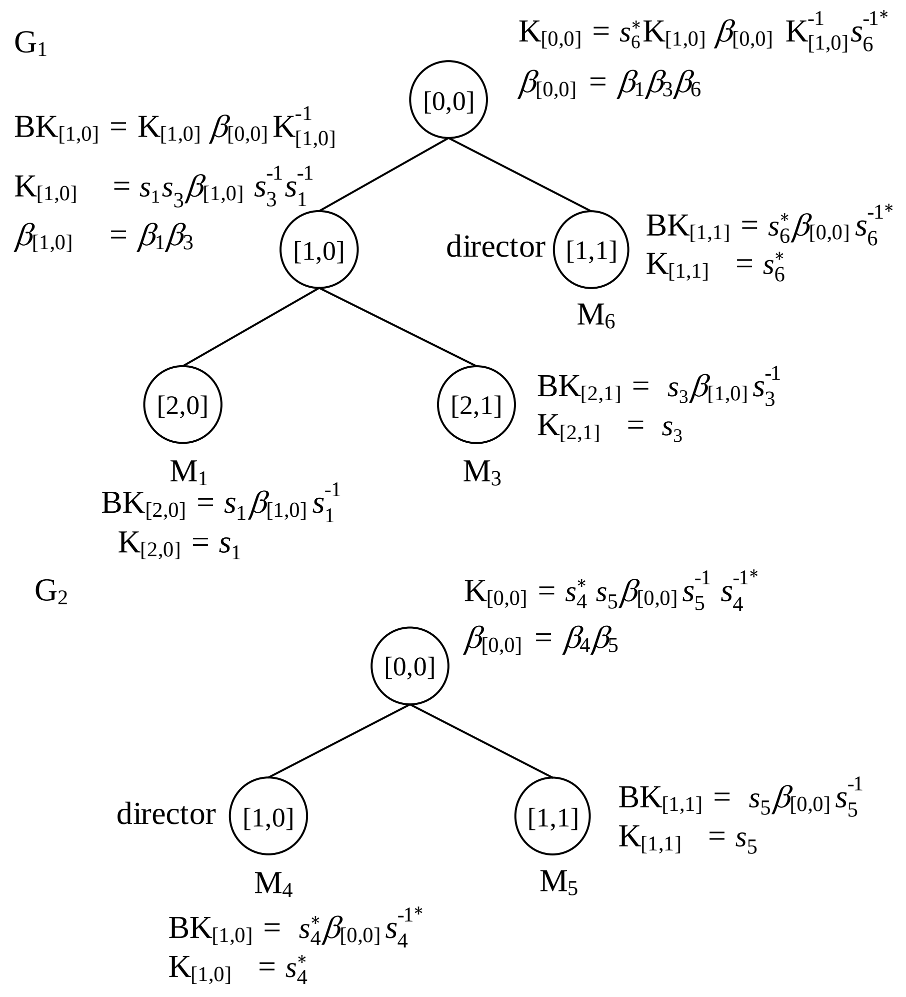

G$_1$
$K_{[0,0]} = s_6^* K_{[1,0]}\, \beta_{[0,0]}\, K_{[1,0]}^{-1} s_6^{-1*}$
$\beta_{[0,0]} = \beta_1 \beta_3 \beta_6$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\, \beta_{[0,0]}\, K_{[1,0]}^{-1}$
$K_{[1,0]} = s_1 s_3 \beta_{[1,0]}\, s_3^{-1} s_1^{-1}$
$\beta_{[1,0]} = \beta_1 \beta_3$
[1,0]
director
[1,1]
$BK_{[1,1]} = s_6^* \beta_{[0,0]}\, s_6^{-1*}$
$K_{[1,1]} = s_6^*$
M$_6$
[2,0]
[2,1]
$BK_{[2,1]} = s_3 \beta_{[1,0]}\, s_3^{-1}$
$K_{[2,1]} = s_3$
M$_1$
M$_3$
$BK_{[2,0]} = s_1 \beta_{[1,0]}\, s_1^{-1}$
$K_{[2,0]} = s_1$
G$_2$
$K_{[0,0]} = s_4^*\, s_5 \beta_{[0,0]}\, s_5^{-1}\, s_4^{-1*}$
$\beta_{[0,0]} = \beta_4 \beta_5$
[0,0]
director
[1,0]
[1,1]
$BK_{[1,1]} = s_5 \beta_{[0,0]}\, s_5^{-1}$
$K_{[1,1]} = s_5$
M$_4$
M$_5$
$BK_{[1,0]} = s_4^* \beta_{[0,0]}\, s_4^{-1*}$
$K_{[1,0]} = s_4^*$

Figure 3.12 After tree updated: Partition Protocol in second scheme

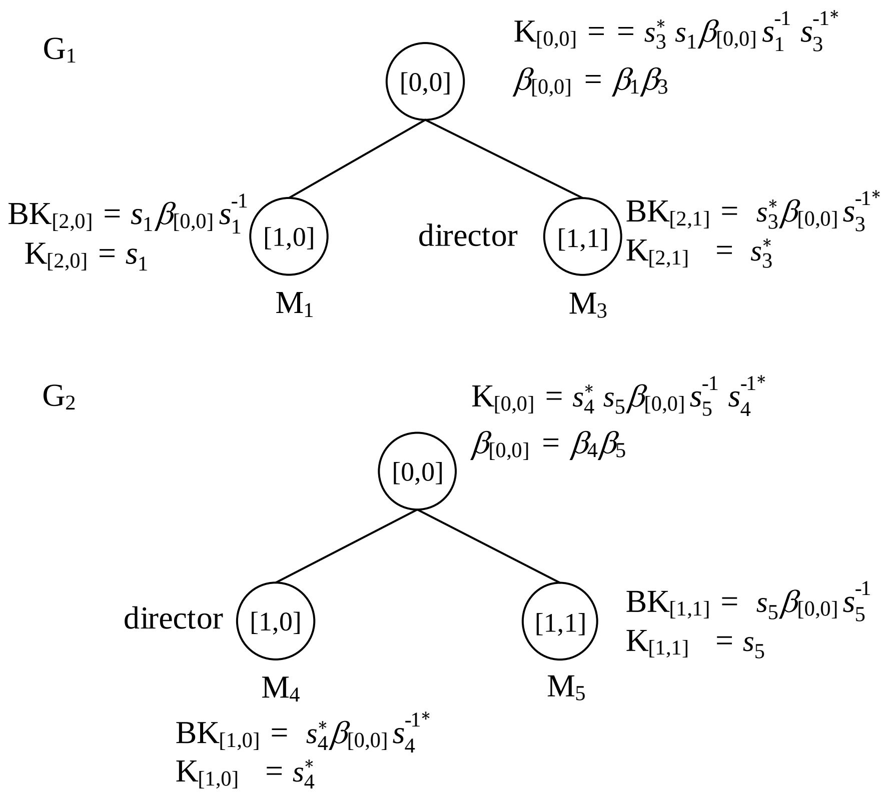

G1
[0,0]
$K_{[0,0]} = = s_3^* s_1 \beta_{[0,0]} s_1^{-1} s_3^{-1*}$
$\beta_{[0,0]} = \beta_1 \beta_3$
$BK_{[2,0]} = s_1 \beta_{[0,0]} s_1^{-1}$
$K_{[2,0]} = s_1$
[1,0]
director
[1,1]
$BK_{[2,1]} = s_3^* \beta_{[0,0]} s_3^{-1*}$
$K_{[2,1]} = s_3^*$
M_1
M_3
G2
$K_{[0,0]} = s_4^* s_5 \beta_{[0,0]} s_5^{-1} s_4^{-1*}$
$\beta_{[0,0]} = \beta_4 \beta_5$
[0,0]
director
[1,0]
[1,1]
$BK_{[1,1]} = s_5 \beta_{[0,0]} s_5^{-1}$
$K_{[1,1]} = s_5$
M_4
M_5
$BK_{[1,0]} = s_4^* \beta_{[0,0]} s_4^{-1*}$
$K_{[1,0]} = s_4^*$

Figure 3.13 After tree updated: Partition Protocol in third scheme

3.3.6 Key Refreshing

Key refreshing in MANET is necessary, since most nodes can be easily compromised due to their mobility and physical vulnerability. Then the key refreshing should be occurred periodically in order to limit exposure due to the loss of keys. Furthermore, the event number limits the amount of ciphertext available to cryptanalysis for given group key. In our protocol the node that needs to refresh the key acts as the director. In similar way of other protocols, the director chooses the new session random key, computes keys and blinded keys up to the root, and broadcasts updated key tree. All members compute the new group key. The conclusion of key refreshing protocol is shown as follows:

Step 1: The director (refreshing node) selects new session random key, computes keys and blinded keys and broadcasts new key tree containing blinded keys.

$$\mathrm{M}_d \xrightarrow{\mathrm{T}^*[\mathrm{BK}]} \{\, \mathrm{M}_i\,,\ i \in [1, n] \,\}$$

Step 2: Each member computes the group key.

3.4 Security Analysis

As described above, it can see that group key agreement on tree-based braid groups satisfies forward and backward secrecy. It also satisfies key independence. The passive adversaries are unable to compute future and previous group key although they know all previous key trees and new key tree respectively, since the director refreshes the session random key every event.

First, the protocol is considered the forward secrecy, note that members that leave the group or passive adversaries who know a contiguous subset of old group keys are unable to compute future group key. The forward secrecy is determined in leave and partition event. Assume *A* as leaving member at position a in key tree *T*. *A* knows all secret keys on key-path that are valid during its group membership. However the director of the leave and partition event updates own session random key and causes the change of keys and blinded keys. Therefore *A* is unable to compute the subsequent group key, because the key tree information is changed. Thus the protocol provides the forward secrecy.

Later, the protocol is considered the backward secrecy to show that new group members are unable to compute old group keys. Assume *A* becomes a new member at position *a* in key tree *T*. As a new member *A* is able to compute all keys on key-path. The director of the join and merge event updates own session random key and causes the change of keys and blinded keys in key-path. Therefore *A* is unable to compute previously used group key, since *A* can only compute new group keys due to changed key tree information. Therefore our protocol satisfies the backward secrecy.

The combination of forward and backward secrecy we follow that TBG protocol satisfies key independence.

3.5 Complexity Analysis

Above section has discussed the security properties of group key agreement schemes. Important is also their complexity, namely performance costs. Sometimes trade-off between complexity and security is required, so that the schemes are suitable to particular environments. Two of the most important criteria are computation costs and communication costs.

(1) Computation Costs

To achieve exact computational costs is impossible and also impracticable. Different implementations of an identical group key agreement scheme bring different results. Even the same implementation cannot guarantee same result in different environments. However, it can estimate the computation costs by identifying the expensive and time-critical operations. The protocol can be ignored the concrete operation time, and only compare the number of such operations. Some computations can be pre-performed before protocol run or computed when the system is idle. Hence only the operations which have to be performed sequentially should be considered. Those operations are denoted by serial operations.

(2) Communication Costs

The communication costs of a group key agreement depend clearly on the topology and properties of the network and the group communication system used. The critical aspects are primarily latency and bandwidth. Additionally the communication costs are implementation-dependent. Hence to achieve fixed communication costs is impracticable. However, the protocol can be estimated them by considering costs including number of rounds, total number of message and number of broadcast and unicast.

Number of rounds: this affects serial communication delay. As the number of rounds grows, the communication delay and the probability of message loss or corruption are increased.

Total number of messages: as the number of messages grows, the probability of message loss or corruption, and the delay are increased.

Number of broadcasts and unicasts: a broadcast operation is much more expensive than a unicast one, since it requires much more acknowledgments with the group communication system. The number of broadcasts should be minimized.

3.5.1 Communication Cost

The communication cost is shown in Table 3.1 that compared among STR, braid groups on GDH and this protocol TBG. The number of rounds on TBG is constant in all events same as STR what better than braid groups on GDH protocols on merge event. The number of rounds on merge operation in Braid groups on GDH depends on number of merging members, but all operations in TBG and STR do not depend on number of members that dynamic movement. The number of rounds in TBG is equals to STR and Braid groups on GDH in join, leave and partition protocol. In merge protocol, the number of rounds in TBG is less than Braid groups on GDH which depending on number of merging member.

3.5.2 Computation Cost

The computation cost in Table 3.2, the serial number of modular exponentiations for STR is O(n). Otherwise the serial number of braid permutations for braid groups on GDH protocol is O(n). For TBG, the serial number of braid permutations is O(n) in leave, merge and partition protocol, except join protocol is constant permutations of braid groups. TBG and Braid groups on GDH reduce the exponential computation in Diffie-Hellman to linear computation by using braid groups.

Table 3.1 Communication Cost of TBG Protocol

Protocol	Operation	Rounds	Message	Unicast Message	Broadcast Message
STR	Join	2	2	1	1
	Leave	1	1	0	1
	Merge	2	3	2	1
	Partition	1	1	0	1
Braid groups on GDH	Join	2	2	1	1
	Leave	1	1	0	1
	Merge	m+3	n+2m+1	n+2m-1	2
	Partition	1	1	0	1
TBG	Join	2	2	1	1
	Leave	1	1	0	1
	Merge	2	2	1	1
	Partition	1	1	0	1

Table 3.2 Computation Cost of TBG Protocol

Protocol	Operation	Exponentiations	Permutation
STR	Join	2	0
	Leave	$3n/2 + 2$	0
	Merge	$2m$	0
	Partition	$3n/2 + 2$	0
Braid groups on GDH	Join	0	$n+3$
	Leave	0	$n-1$
	Merge	0	$n+2m+1$
	Partition	0	$n-p$
TBG	Join	0	n
	Leave	0	$n-1$
	Merge	0	$n+m$
	Partition	0	$n-p$

CHAPTER 4

AUTHENTICATED GROUP KEY AGREEMENT USING TREE-BASED BRAID GROUPS

This protocol uses public and authentic channel, it means that everyone, both the member and the adversary, can read the messages. Man-in-the-middle attack works on TBG protocol in chapter 3. The authenticated process can resistant to them. Public key infrastructure (PKI) will be used if it exists. However, unlike in traditional networks, no present PKI can be assumed in ad hoc networks. Therefore the authenticated group key agreement on MANET using tree-based braid groups is introduced in this chapter for solving the man-in-the-middle attack. The notations in this protocol are denoted as follows:

n	number of protocol participants (group members)
m	number of merging members
i ,r, d	indices of group members
M_i	ith group member; $i \in \{1, 2, \cdot\cdot\cdot, n\}$
M_*	all group members
s_i	session random key of M_i from subgroups (B_{g_i}) of B_g
x_i	long term private key of M_i
P_i	long term public key of M_i
h	height of tree
T	key tree
T^*	tree after membership operation
$K_{[h, v]}$	secret key at [h, v] node
$BK_{[h, v]}$	blinded key at [h, v] node
[h, v]	v^{th} node at level h in a tree

The initial assumption of authentication scheme is explained as follows.

M$_i$'s long-term private key

$$x_i \in B_{g_i}$$

M$_i$'s long-term public key

$$P_i = x_i \alpha x_i^{-1}$$

where $\alpha \in B_g$ is a published braid what be a sufficiently complicated braid.

Therefore, the long-term public value of the group are $\{(B_{gi}, \alpha, x_i \alpha x_i^{-1}) \mid i = 1,..., n\}$.The long-term public key of each member in the protocol is guaranteed by some trust party such as off-line CA before group operation.

4.1 Two-party Key Agreement Protocol

Following the above mentioned notations, the authenticated two party key agreement protocol based on Chaturvedi et al (2008) is described below. The protocol works in the following steps.

Message 1 :

$$\begin{tikzcd}\text{Alice} \arrow[r, "k_a BK_a k_a^{-1}"] & \text{Bob}\end{tikzcd}$$

Message 2 :

$$\begin{tikzcd}\text{Alice} & \text{Bob} \arrow[l, "k_b BK_b k_b^{-1}"']\end{tikzcd}$$

Figure 4.1 Two-party key agreement protocol

For message 1 : Alice challenges Bob

Step 1: When Alice wants to share the key to Bob, Alice selects a session random key $s_a \in B_{g_a}$ to achieve $BK_a = s_a \beta s_a^{-1}$ and computes $k_a = x_a P_b x_a^{-1}$, where x_a is

Alice's long term private key and P_b is Bob's long-term public key. Then Alice sends the message, authenticated blinded key, through Bob with

$$k_a BK_a k_a^{-1} = x_a P_b x_a^{-1} s_a \beta s_a^{-1} [x_a P_b x_a^{-1}]^{-1},$$

where $\beta \in B_g$ is a published braid what be a sufficiently complicated braid.

Step 2: Bob computes Alice's blinded key BK_a as follows:

$$\begin{aligned} BK_a &= k_b^{-1} k_a\, BK_a\, k_a^{-1} k_b \\ &= [x_b P_a x_b^{-1}]^{-1} [x_a P_b x_a^{-1}]\, s_a \beta s_a^{-1} [x_a P_b x_a^{-1}]^{-1} [x_b P_a x_b^{-1}] \\ &= [x_b x_a \alpha x_a^{-1} x_b^{-1}]^{-1} [x_a x_b \alpha x_b^{-1} x_a^{-1}]\, s_a \beta s_a^{-1} [x_a x_b \alpha x_b^{-1} x_a^{-1}]^{-1} [x_b x_a \alpha x_a^{-1} x_b^{-1}] \\ &= s_a \beta s_a^{-1} \end{aligned}$$

where Bob computes $k_b = x_b P_a x_b^{-1}$. Then the Alice's blinded key what be computed by Bob is $s_a \beta s_a^{-1}$.

Step 3: The Bob compute shared key as $s_b s_a \beta s_a^{-1} s_b^{-1}$ where Bob's session random key $s_b \in B_{g_b}$.

Moreover Alice can compute shared key in the similar step as follows:

For message 2 : Bob challenges Alice

Step 1 When Bob wants to share the key to Alice, Bob selects a session random key $s_b \in B_{g_b}$ to achieve $BK_b = s_b \beta s_b^{-1}$ and computes $k_b = x_b P_a x_b^{-1}$, where where x_b is Bob's long term private key and P_a is Alice's long-term public key. Then Bob sends the message, authenticated blinded key, through Alice with

$$k_b BK_b k_b^{-1} = x_b P_a x_b^{-1} s_b \beta s_b^{-1} [x_b P_a x_b^{-1}]^{-1},$$

where $\beta \in B_g$ is a published braid what be a sufficiently complicated braid.

Step 2: Alice computes Bob's blinded key BK_b as follows:

$BK_b = k_a^{-1} k_b \, BK_b \, k_b^{-1} k_a$

$= [x_a P_b x_a^{-1}]^{-1} [x_b P_a x_b^{-1}] \, s_b \beta s_b^{-1} [x_b P_a x_b^{-1}]^{-1} [x_a P_b x_a^{-1}]$

$= [x_a x_b \alpha x_b^{-1} x_a^{-1}]^{-1} [x_b x_a \alpha x_a^{-1} x_b^{-1}] \, s_b \beta s_b^{-1} [x_b x_a \alpha x_a^{-1} x_b^{-1}]^{-1} [x_a x_b \alpha x_b^{-1} x_a^{-1}]$

$= s_b \beta s_b^{-1}$

where Alice computes $k_a = x_a P_b x_a^{-1}$. Then the Bob's blinded key what be computed by Alice is $s_b \beta s_b^{-1}$.

Step 3: The Alice computes shared key as $s_a s_b \beta s_b^{-1} s_a^{-1}$ where Alice's session random key $s_a \in B_{g_a}$.

Therefore, after the regular protocol running, the Alice and Bob achieve same shared key because of $s_a s_b \beta s_b^{-1} s_a^{-1} = s_b s_a \beta s_a^{-1} s_b^{-1}$.

4.2 Authenticated Group Key Agreement Protocol on Tree-based Braid Groups (ATBG)

This section describes how ATBG deals with the group operations including setup, join, leave, merge, partition and refresh. The protocol is based on the authentication phase according to section 4.1.

4.2.1 Setup Protocol

The members who want to form a group can be ordered according to some criteria such as MAC address of device same as the previous protocol. The structure of the key tree can be then derived from this order. The first member in the order is selected as director. The blinded key of member M_i is $BK_i = s_i \beta_r \beta_i s_i^{-1}$ where β_r is existing publish braid word at root node before the director will update next member to key tree by order. Each member knows the own β_r because it has some criteria such as MAC address of all members. It can order the MAC address by itself, and then it knows sequence of member. Each member sends its authenticated blinded key according to director. For example, M_5 sends authenticated blinded key as $k_5 BK_5 k_5^{-1}$ where k_5 is $x_5 P_d x_5^{-1}$, x_5 is M_5's long term private key and P_d is director's long term public key. The process is illustrated as follows:

Step 1: Each M_i, $i \in \{1, \cdot \cdot \cdot , n\}$ sends its authenticated blinded session random key to director M_d.

$$\{ M_i , i \in [1, n] \} - M_d \xrightarrow{k_{i\_d}\, BK_i k_{i\_d}^{-1}} M_d$$

where $k_{i\_d} = x_i P_d x_i^{-1}$, x_i is M_i's long term private key and P_d is M_d's long term public key.

Step 2: The director computes the member blinded keys from $k_{d\_i}^{-1} k_{i\_d} BK_i k_{i\_d}^{-1} k_{d\_i}$ to achieve BK_i, where $k_{d\_i} = x_d P_i x_d^{-1}$.

Step 3:The director creates key tree and computes keys and blinded keys to the root.

Step 4: The director unicasts key tree with authenticated blinded keys to each member.

$$M_d \xrightarrow{k_{d\_i} T[BK]\, k_{d\_i}^{-1}} \{ M_i , i \in [1, n] \} - M_d$$

Step 5: Each member computes authenticated blinded keys as $k_{i\_d}^{-1} k_{d\_i} T[BK]\ k_{d\_i}^{-1} k_{i\_d}$ to achieve the blinded keys in key tree, T[BK], where $k_{i\_d} = x_i P_d x_i^{-1}$.

Step 6: Each member computes the secret group key.

Then total communication message in setup protocol is n rounds including the authenticated blinded key from each member to director in n – 1 rounds and authenticated key tree information from director to all members in one round. The number of unicast message is 2(n – 1) including the authenticated blinded key from each member to director in n – 1 messages and authenticated key tree information from director to all members in n – 1 messages.

4.2.2 Join Protocol

The current group has n members, the new member is identified with M_{n+1}. The tree will be added a new intermediate node with two children: the root node of the prior tree on the left and the new leaf node on the right for new member. This node becomes the new root node. As mention in section 3.3.2, the director who is the maximum signal strength with joining member is selected from current group members. For simplicity, the protocol use n in the following to denote the number of group members before operation join. To deal with the join operation, a member is chosen as the director, because the new member detects the maximum signal strength. The new member M_{n+1} sends a join request, JOIN_MESSAGE, to the director. Later the director refreshes the session random key, computes keys and blinded keys of intermediate nodes up to the root node and sends authenticated blinded keys in key tree to M_{n+1}. Next, the member M_{n+1} computes the blinded keys in key tree and updates existing key tree in accordance with creates a new root node and a new member node. Next, the new member selects session random key and computes keys and blinded keys going up to the root. The blinded key of new member M_{n+1} is $BK_{n+1} = s_{n+1}\beta_r\beta_{n+1}s_{n+1}^{-1}$ where β_r is existing publish braid word at root node that the new member can find in existing key tree information. The new member unicasts the new key tree containing only authenticated blinded keys to all other members. Finally, each member computes blinded keys and group key. This join protocol provides key independence since director updates session random key that knowledge of a previous group key cannot be used to compute the new group key. Figure 4.1 shows the authenticated key tree what director, M_1, sends to new member, M_4. This instance, it means that the M_1 is nearest with M_4. Figure 4.2 shows the authenticated key tree that M_2 received from new member, M_4. Figure 4.3 shows key tree information after M_1 computes keys and blinded keys. The summary process of protocol is illustrated as follows:

Step 1: The new member detects the maximum signal strength of current group members as director and sends JOIN_MESSAGE request message to join the group. After the director receives the request message, it selects its new session random key, computes keys and blinded keys, and sends the existing authenticated key tree to new member.

$$M_d \xrightarrow{k_{d\_n+1} T[BK]\, k^{-1}_{d\_n+1}} M_{n+1}$$

where $k_{d\_n+1} = x_d P_{n+1} x_d^{-1}$, x_d is director's long term private key and P_{n+1} is new member's long term public key.

Step 2: The new member computes the existing blinded keys from $k^{-1}_{n+1\_d}\, k_{d\_n+1}\, T^*[BK]\, k^{-1}_{d\_n+1}\, k_{n+1\_d}$ to achieve blinded keys in key tree, where $k_{n+1\_d} = x_{n+1} P_d x^{-1}_{n+1}$.

Step 3: The new member selects its session random key, updates key tree and computes keys and blinded keys.

Step 4: The new member unicasts the new key tree with authenticated blinded keys to all member.

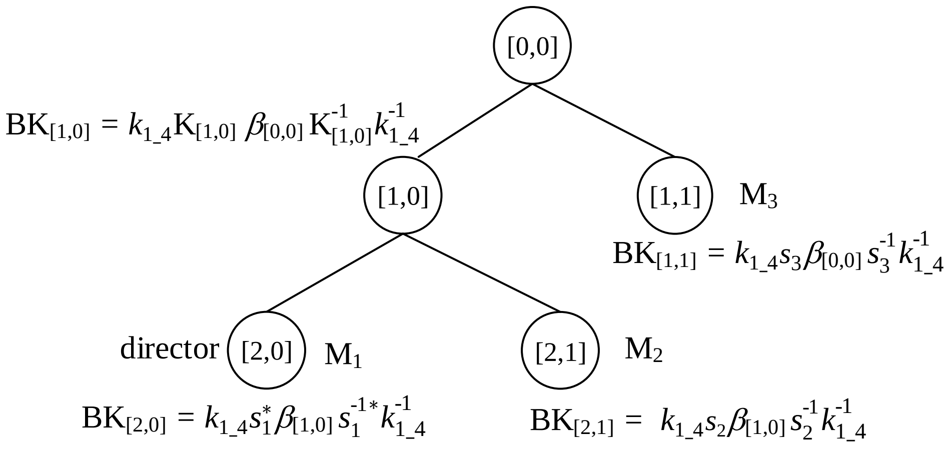
$$M_{n+1} \xrightarrow{k_{n+1\_i} T^*[BK]\, k^{-1}_{n+1\_i}} \{ M_i , i \in [1, n] \}$$

where $k_{n+1\_i} = x_{n+1} P_i x^{-1}_{n+1}$, x_{n+1} is new member's long term private key and P_i is M_i's long term public key.

Step 5: Each member computes authenticated blinded keys as $k^{-1}_{i\_n+1}\, k_{n+1\_i}\, \mathrm{T}^{*}[\mathrm{BK}]\, k^{-1}_{n+1\_i}\, k_{i\_n+1}$ to achieve the blinded keys in new key tree, T^{*}[BK], where $k_{i\_n+1} = x_i\, \mathrm{P}_{n+1}\, x_i^{-1}$.

Step 6: Each member computes the secret group key.

Then total communication message in setup protocol is two rounds including existing authenticated key tree information from director to new member in one round and new key tree information from new member to each member in one round. The total amount of message is *n* + 1 including information from director to new member in one message and information that is unicasted by new member to current group members before change operation in *n* messages.



[0,0]
$\mathrm{BK}_{[1,0]} = k_{1\_4}\mathrm{K}_{[1,0]}\,\beta_{[0,0]}\,\mathrm{K}^{-1}_{[1,0]}k^{-1}_{1\_4}$
[1,0]
[1,1]
M3
$\mathrm{BK}_{[1,1]} = k_{1\_4}s_3\beta_{[0,0]}\,s_3^{-1}k^{-1}_{1\_4}$
director
[2,0]
M1
[2,1]
M2
$\mathrm{BK}_{[2,0]} = k_{1\_4}s_1^{*}\beta_{[1,0]}\,s_1^{-1*}k^{-1}_{1\_4}$
$\mathrm{BK}_{[2,1]} = k_{1\_4}s_2\beta_{[1,0]}\,s_2^{-1}k^{-1}_{1\_4}$

Figure 4.1 Join protocol : Updated tree that new member received from director

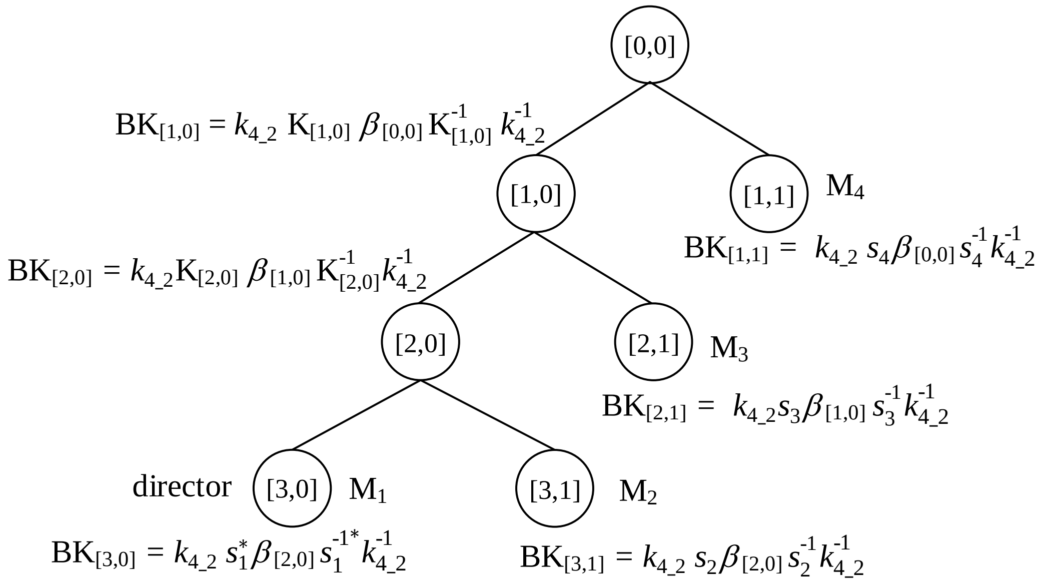

[0,0]
$BK_{[1,0]} = k_{4\_2}\, K_{[1,0]}\, \beta_{[0,0]}\, K^{-1}_{[1,0]}\, k^{-1}_{4\_2}$
[1,0]
[1,1]
M₄
$BK_{[1,1]} = k_{4\_2}\, s_4 \beta_{[0,0]}\, s_4^{-1} k^{-1}_{4\_2}$
$BK_{[2,0]} = k_{4\_2} K_{[2,0]}\, \beta_{[1,0]}\, K^{-1}_{[2,0]} k^{-1}_{4\_2}$
[2,0]
[2,1]
M₃
$BK_{[2,1]} = k_{4\_2} s_3 \beta_{[1,0]}\, s_3^{-1} k^{-1}_{4\_2}$
director
[3,0]
M₁
[3,1]
M₂
$BK_{[3,0]} = k_{4\_2}\, s_1^* \beta_{[2,0]}\, s_1^{-1*} k^{-1}_{4\_2}$
$BK_{[3,1]} = k_{4\_2}\, s_2 \beta_{[2,0]}\, s_2^{-1} k^{-1}_{4\_2}$

Figure 4.2 Join protocol : Authenticated key tree that M₂ received from new member, M₄

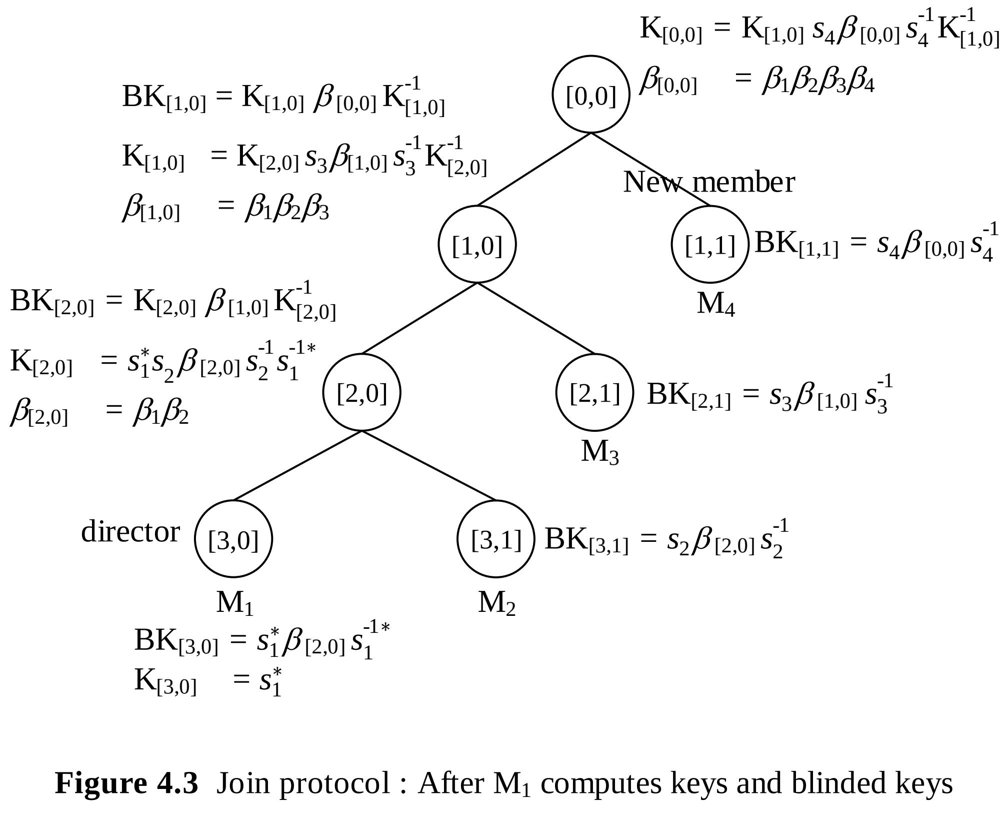

$K_{[0,0]} = K_{[1,0]}\, s_4 \beta_{[0,0]}\, s_4^{-1} K^{-1}_{[1,0]}$
$\beta_{[0,0]} = \beta_1\beta_2\beta_3\beta_4$
$BK_{[1,0]} = K_{[1,0]}\, \beta_{[0,0]}\, K^{-1}_{[1,0]}$
[0,0]
$K_{[1,0]} = K_{[2,0]}\, s_3 \beta_{[1,0]}\, s_3^{-1} K^{-1}_{[2,0]}$
$\beta_{[1,0]} = \beta_1\beta_2\beta_3$
New member
[1,0]
[1,1]
$BK_{[1,1]} = s_4 \beta_{[0,0]}\, s_4^{-1}$
M₄
$BK_{[2,0]} = K_{[2,0]}\, \beta_{[1,0]}\, K^{-1}_{[2,0]}$
$K_{[2,0]} = s_1^* s_2 \beta_{[2,0]}\, s_2^{-1} s_1^{-1*}$
$\beta_{[2,0]} = \beta_1\beta_2$
[2,0]
[2,1]
$BK_{[2,1]} = s_3 \beta_{[1,0]}\, s_3^{-1}$
M₃
director
[3,0]
[3,1]
$BK_{[3,1]} = s_2 \beta_{[2,0]}\, s_2^{-1}$
M₁
M₂
$BK_{[3,0]} = s_1^* \beta_{[2,0]}\, s_1^{-1*}$
$K_{[3,0]} = s_1^*$

Figure 4.3 Join protocol : After M₁ computes keys and blinded keys

4.2.3 Leave Protocol

Like in TBG, the leave protocol is relatively simple, only one round is needed. Suppose a group has n members and the member M_r with $1 \leq r \leq n$ leaves the group. Again the protocol needs the director M_d to updates its session random key. If the leaving node is child of root, the director is leaf node directly below the removing member, otherwise the director is the leaf node directly above the removing member in existing key tree before leave event. After notification of the leave event from the group communication system, the director refreshes a secret session random key, updates the key tree by deleting the nodes of leaving member and computes keys and blinded keys going up to the root. The director computes the new blinded key of leaf nodes below and above removing node same as leave protocol in section 3.3.3. Next, the director unicasts the new key tree containing only authenticated blinded keys to remaining member. Figure 4.4 shows situation before a member, M_1, leaves. After that Figure 4.5 shows the authenticated key tree that M_3 received from director, M_2, after the example of M_1 leaving a group. Figure 4.6 shows key tree information after M_4 computes keys and blinded keys. The conclusion of leave protocol is illustrated as follows:

Step 1: The director selects the new session random key, updates the key tree and computes keys and blinded keys.

Step 2: The director unicasts the new authenticated key tree to each member except leaving member.

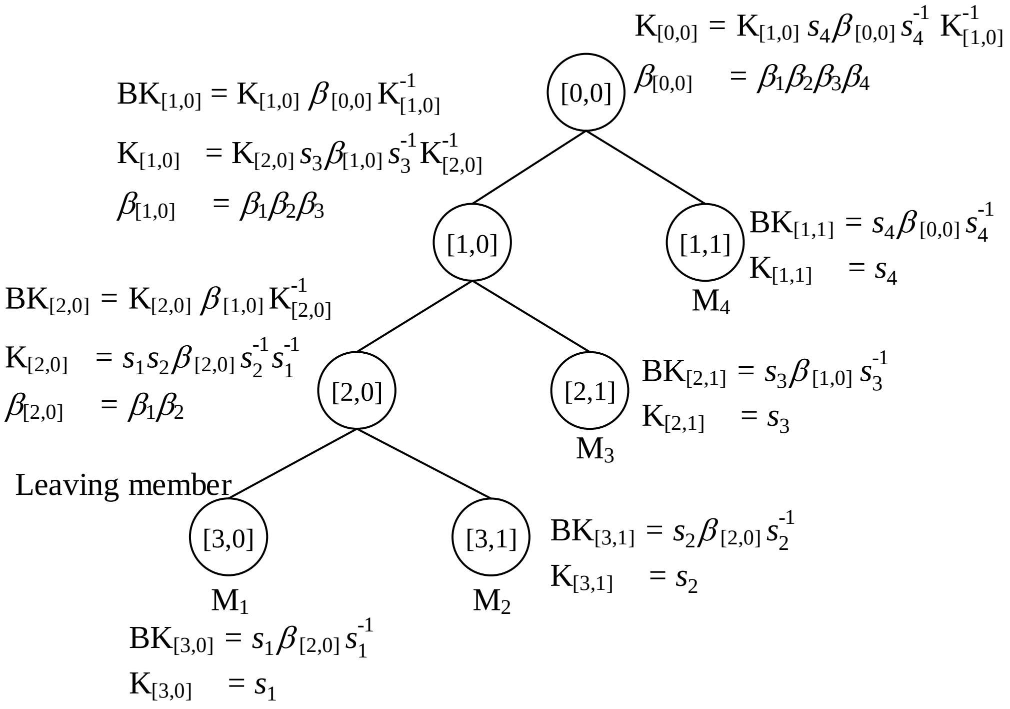
$$M_d \xrightarrow{k_{d\_i} T^{*}[BK]\, k_{d\_i}^{-1}} \{ M_i ,\, i \in [1, n] \} - M_r$$

where $k_{d\_i} = x_d P_i x_d^{-1}$, x_d is director's long term private key and P_i is M_i's long term public key.

Step 3: Each member computes authenticated blinded keys as $k_{i\_d}^{-1}\, k_{d\_i}\, T^{*}[BK]\, k_{d\_i}^{-1}\, k_{i\_d}$ to achieve the blinded keys in new key tree, T^{*}[BK], where $k_{i\_d} = x_i\, P_d\, x_i^{-1}$.

Step 4: Each member computes the secret group key.

Then total communication message in leave protocol is one round that director unicasts message to each member. Then the total amount of unicast message is n-1 including the messages that director unicasts information to remaining group members except leaving member.



$K_{[0,0]} = K_{[1,0]}\, s_4 \beta_{[0,0]}\, s_4^{-1}\, K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_1\beta_2\beta_3\beta_4$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\, \beta_{[0,0]}\, K_{[1,0]}^{-1}$
$K_{[1,0]} = K_{[2,0]}\, s_3\beta_{[1,0]}\, s_3^{-1} K_{[2,0]}^{-1}$
$\beta_{[1,0]} = \beta_1\beta_2\beta_3$
[1,0]
[1,1]
M_4
$BK_{[1,1]} = s_4\beta_{[0,0]}\, s_4^{-1}$
$K_{[1,1]} = s_4$
$BK_{[2,0]} = K_{[2,0]}\, \beta_{[1,0]}\, K_{[2,0]}^{-1}$
$K_{[2,0]} = s_1 s_2 \beta_{[2,0]}\, s_2^{-1} s_1^{-1}$
$\beta_{[2,0]} = \beta_1\beta_2$
[2,0]
[2,1]
M_3
$BK_{[2,1]} = s_3\beta_{[1,0]}\, s_3^{-1}$
$K_{[2,1]} = s_3$
Leaving member
[3,0]
[3,1]
M_1
M_2
$BK_{[3,1]} = s_2\beta_{[2,0]}\, s_2^{-1}$
$K_{[3,1]} = s_2$
$BK_{[3,0]} = s_1\beta_{[2,0]}\, s_1^{-1}$
$K_{[3,0]} = s_1$

Figure 4.4 Leave protocol : Before tree updated that M_1 leaves

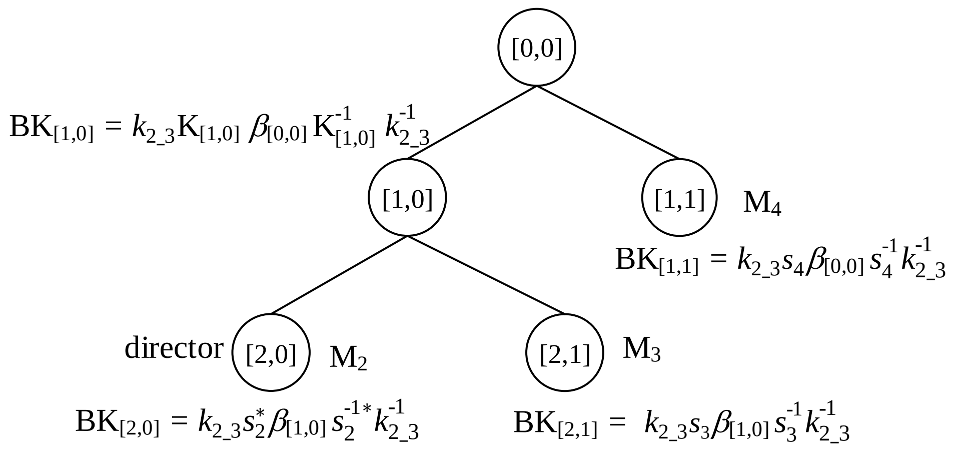

[0,0]
$BK_{[1,0]} = k_{2\_3}K_{[1,0]}\beta_{[0,0]}K^{-1}_{[1,0]}k^{-1}_{2\_3}$
[1,0]
[1,1]
M4
$BK_{[1,1]} = k_{2\_3}s_4\beta_{[0,0]}s^{-1}_4k^{-1}_{2\_3}$
director
[2,0]
M2
[2,1]
M3
$BK_{[2,0]} = k_{2\_3}s^*_2\beta_{[1,0]}s^{-1*}_2k^{-1}_{2\_3}$
$BK_{[2,1]} = k_{2\_3}s_3\beta_{[1,0]}s^{-1}_3k^{-1}_{2\_3}$

Figure 4.5 Leave protocol : Authenticated key tree that M_3 receives from director, M_2

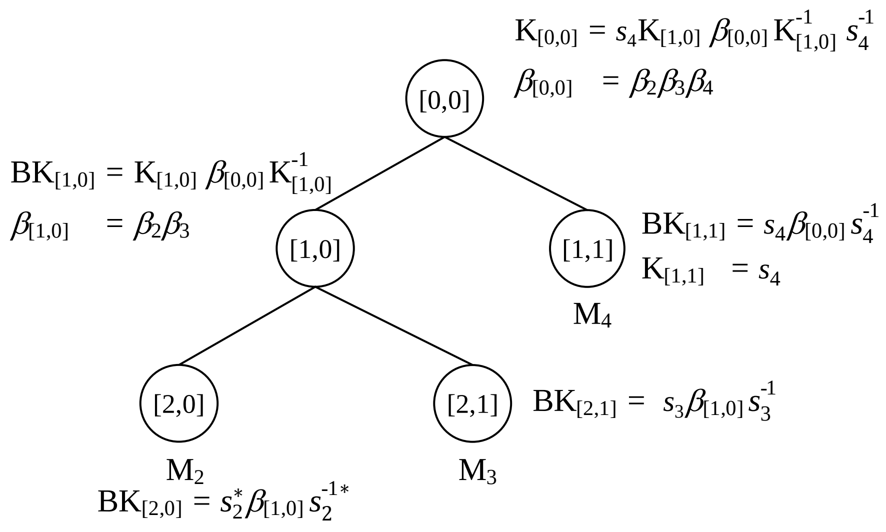

$K_{[0,0]} = s_4K_{[1,0]}\beta_{[0,0]}K^{-1}_{[1,0]}s^{-1}_4$
$\beta_{[0,0]} = \beta_2\beta_3\beta_4$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\beta_{[0,0]}K^{-1}_{[1,0]}$
$\beta_{[1,0]} = \beta_2\beta_3$
[1,0]
[1,1]
$BK_{[1,1]} = s_4\beta_{[0,0]}s^{-1}_4$
$K_{[1,1]} = s_4$
M4
[2,0]
[2,1]
$BK_{[2,1]} = s_3\beta_{[1,0]}s^{-1}_3$
M2
M3
$BK_{[2,0]} = s^*_2\beta_{[1,0]}s^{-1*}_2$

Figure 4.6 Leave protocol : After M_4 computes keys and blinded keys

4.2.4 Merge Protocol

Like the TBG in previous protocol, this protocol assumes that *m* merging group needs to merge with *c* current group. The existing merging group director detects to achieve maximum signal strength what is measured as the closest member between itself and current group members. The current group director is the member that has maximum signal strength with merging group director. After the merging process, the smaller group is merged onto the larger one, i.e. to place a smaller key tree directly on top of the larger one. If group sizes are equal, it can order them according to some other criteria. A new intermediate node with two children is created. The root of the larger tree becomes the left child of new intermediate node, while the deepest leaf of the smaller tree the right child of new intermediate node. The root of smaller tree becomes the root of the new tree.

After the current group director receives the MERGE_MESSAGE message, it refreshes session random key, computes keys and blinded keys, and sends the current group's key tree containing the all authenticated blinded keys to merging group director. Later, the merging group director computes blinded key of current group's key tree, updates key tree by combining the merging group's key tree and current group's key tree at the new root node, the director chooses session random key, computes keys and blinded keys up to the root node, and unicasts new key tree containing the all authenticated blinded keys to all members in new group. Finally, the group key is calculated independently by each member after computed blinded key. Figure 4.7 shows the initial situation before current group merges with other group. The member that has maximum signal strength of merging group director, M_6, is M_4, and then M_4 is current group director. After that Figure 4.8 shows the authenticated key tree that merging director, M_6, received from current group director, M_4. Figure 4.9 shows the authenticated key tree that M_3 received from merging group director, M_6, after merging group director merges the key tree. Figure 4.10 shows the key tree after M_2 computes keys and blinded keys. The conclusion of merge protocol is shown as follows:

Step 1: The current group director that selects new session random key, compute blinded keys and sends update key tree with authenticated blinded key to the merging group director.

$$M_{d_c} \xrightarrow{k_{d_c\_d_m} T^*_{d_c}[BK] k^{-1}_{d_c\_d_m}} M_{d_m}$$

where $k_{d_c\_d_m} = x_{d_c} P_{d_m} x_{d_c}^{-1}$, x_{d_c} is current group director's long term private key, P_{d_m} is merging group director's long term public key and $T_{d_c}^{*}[BK]$ is key tree of current group with new session random key of current group director.

Step 2: The merging group director computes the current group blinded key from$k_{d_m_d_c}^{-1} k_{d_c_d_m} T_{d_c}^{*}[BK]\ k_{d_c_d_m}^{-1} k_{d_m_d_c}$ to achieve blinded keys in key tree, where $k_{d_m\_d_c} = x_{d_m} P_{d_c}\ x_{d_m}^{-1}$.

Step 3: The merging group director selects its new session random key, combines key tree, computes keys and blinded keys.

Step 4: The merging group director unicasts new key tree with authenticated blinded keys to all members.

$$M_{d_m} \xrightarrow{k_{d_m\_i} T^{*}[BK] k_{d_m\_i}^{-1}} \{\ M_i\ ,\ i \in [1, n+m]\ \}$$

where $k_{d_m\_i} = x_{d_m} P_i\ x_{d_m}^{-1}$, x_{d_m} is merging group director's long term private key and P_i is M_i's long term public key.

Step 5: Each member computes authenticated blinded key as $k_{i\_d_m}^{-1} k_{d_m\_i} T^{*}[BK]\ k_{d_m\_i}^{-1}\ k_{i\_d_m}$ to achieve the blinded keys in new key tree, $T^{*}[BK]$, where $k_{i\_d_m} = x_i P_{d_m}\ x_i^{-1}$.

Step 6: Each member computes the secret group key.

Therefore the total communication message in leave protocol is two rounds including current group director sends authenticated key tree to merging group director in one round and merging group director unicasts authenticated new key tree to each member in one round. The total amount of message is n + m including the messages that current group director sends information to merging group director in one message and merging group director sends authenticated key tree to each group member in $n + m - 1$ messages.

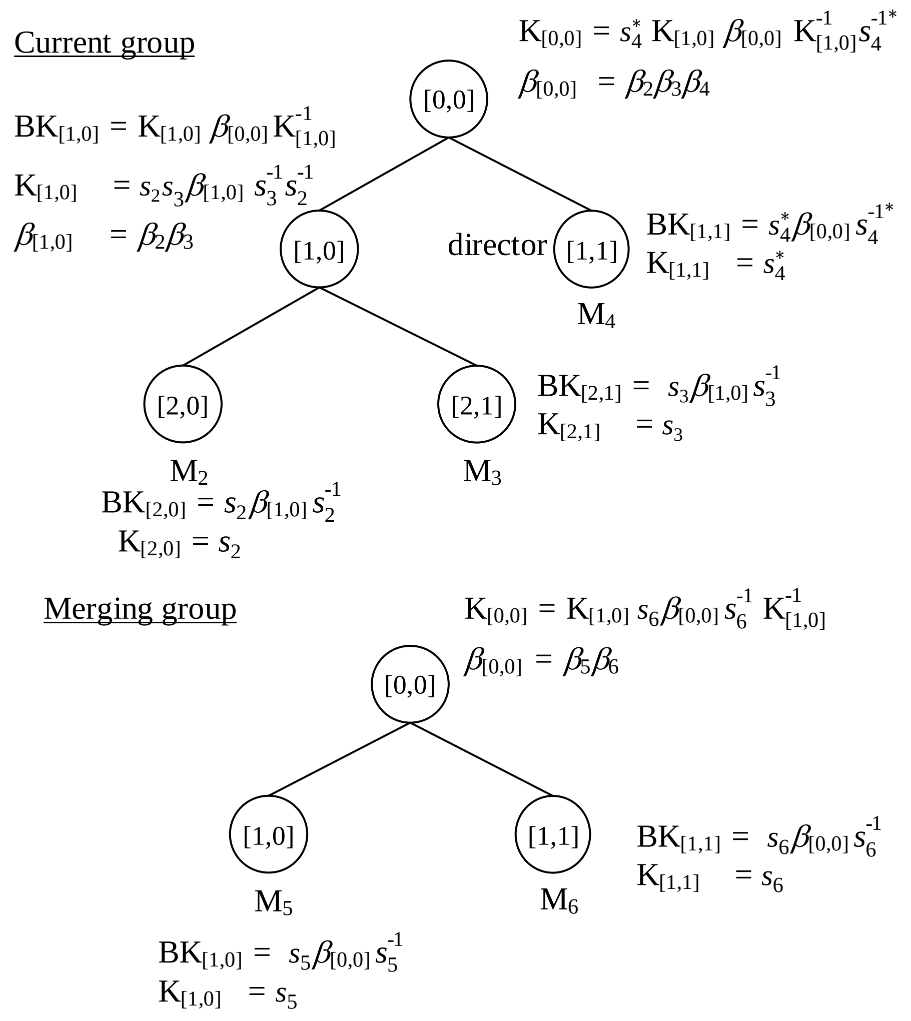

Current group
$K_{[0,0]} = s_4^* K_{[1,0]} \beta_{[0,0]} K_{[1,0]}^{-1} s_4^{-1*}$
$\beta_{[0,0]} = \beta_2\beta_3\beta_4$
[0,0]
$BK_{[1,0]} = K_{[1,0]} \beta_{[0,0]} K_{[1,0]}^{-1}$
$K_{[1,0]} = s_2 s_3 \beta_{[1,0]} s_3^{-1} s_2^{-1}$
$\beta_{[1,0]} = \beta_2\beta_3$
[1,0]
director
[1,1]
M_4
$BK_{[1,1]} = s_4^* \beta_{[0,0]} s_4^{-1*}$
$K_{[1,1]} = s_4^*$
[2,0]
[2,1]
$BK_{[2,1]} = s_3 \beta_{[1,0]} s_3^{-1}$
$K_{[2,1]} = s_3$
M_2
M_3
$BK_{[2,0]} = s_2 \beta_{[1,0]} s_2^{-1}$
$K_{[2,0]} = s_2$
Merging group
$K_{[0,0]} = K_{[1,0]} s_6 \beta_{[0,0]} s_6^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_5\beta_6$
[0,0]
[1,0]
[1,1]
$BK_{[1,1]} = s_6 \beta_{[0,0]} s_6^{-1}$
$K_{[1,1]} = s_6$
M_5
M_6
$BK_{[1,0]} = s_5 \beta_{[0,0]} s_5^{-1}$
$K_{[1,0]} = s_5$

Figure 4.7 Merge Protocol : Before merging process

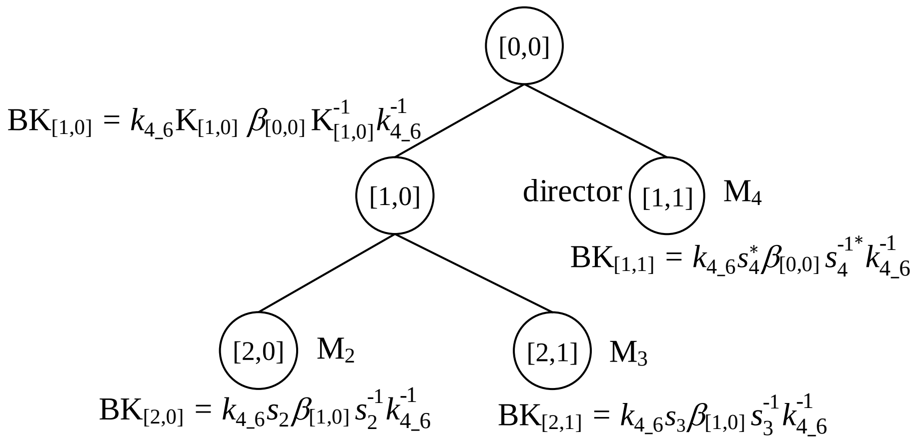

[0,0]
$BK_{[1,0]} = k_{4\_6}K_{[1,0]}\,\beta_{[0,0]}\,K^{-1}_{[1,0]}k^{-1}_{4\_6}$
[1,0]
director
[1,1]
M4
$BK_{[1,1]} = k_{4\_6}s^{*}_{4}\beta_{[0,0]}\,s^{-1*}_{4}\,k^{-1}_{4\_6}$
[2,0]
M2
[2,1]
M3
$BK_{[2,0]} = k_{4\_6}s_{2}\beta_{[1,0]}\,s^{-1}_{2}k^{-1}_{4\_6}$
$BK_{[2,1]} = k_{4\_6}s_{3}\beta_{[1,0]}\,s^{-1}_{3}k^{-1}_{4\_6}$

Figure 4.8 Merge protocol : Authenticated key tree that merging director, M_6, received from current group director, M_4

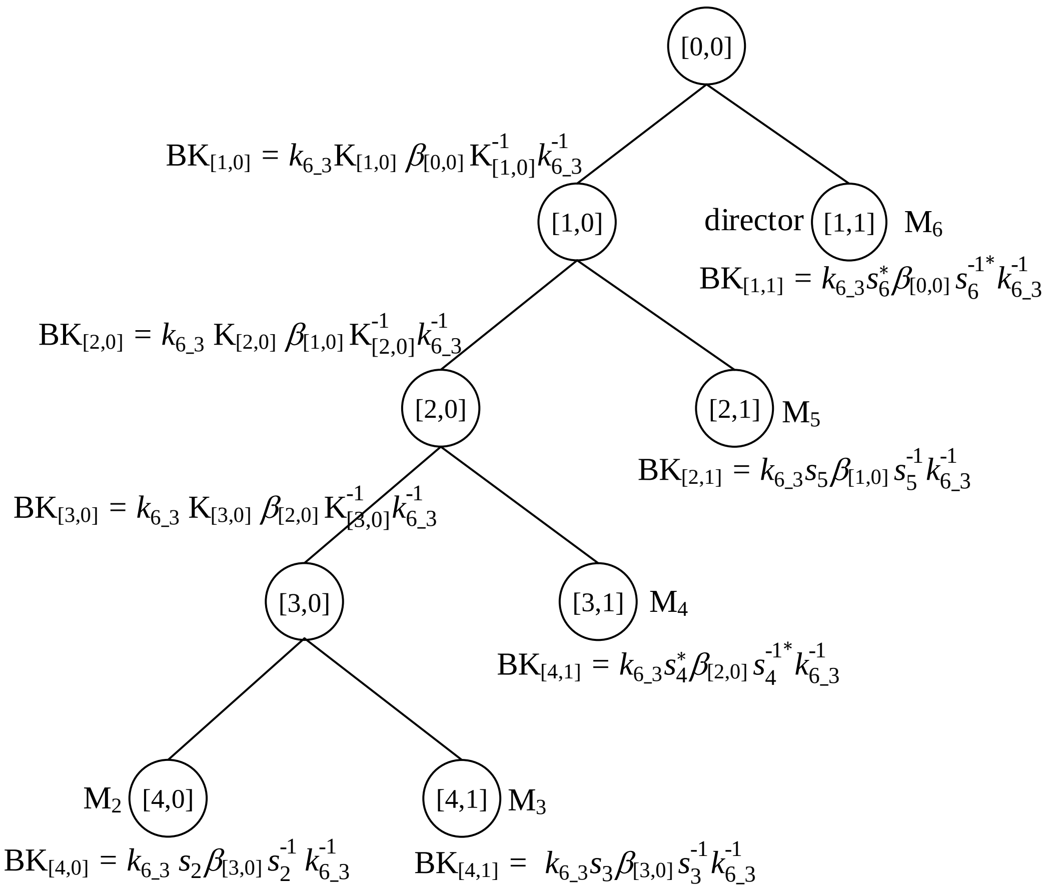
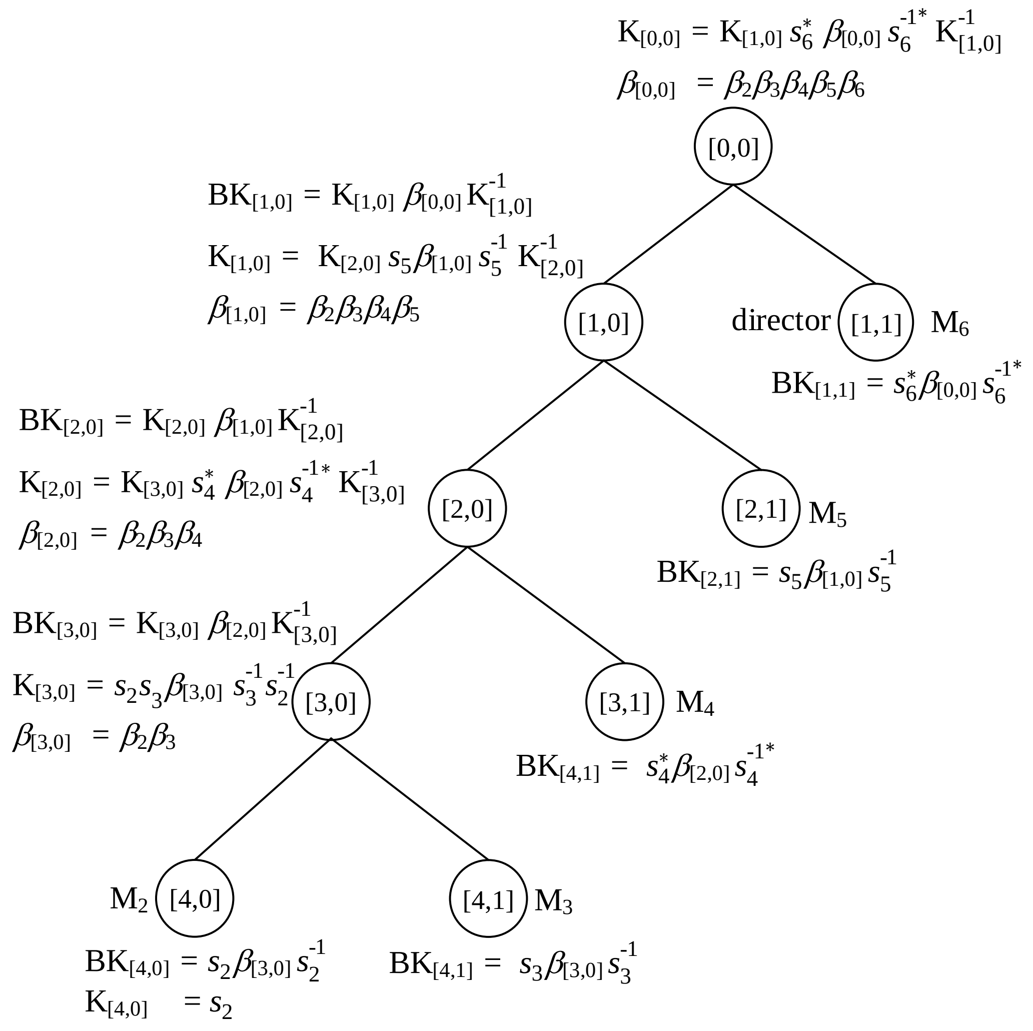

[0,0]
$BK_{[1,0]} = k_{6\_3}K_{[1,0]}\,\beta_{[0,0]}\,K^{-1}_{[1,0]}k^{-1}_{6\_3}$
[1,0]
director
[1,1]
M6
$BK_{[1,1]} = k_{6\_3}s^{*}_{6}\beta_{[0,0]}\,s^{-1*}_{6}\,k^{-1}_{6\_3}$
$BK_{[2,0]} = k_{6\_3}\,K_{[2,0]}\,\beta_{[1,0]}\,K^{-1}_{[2,0]}k^{-1}_{6\_3}$
[2,0]
[2,1]
M5
$BK_{[2,1]} = k_{6\_3}s_{5}\beta_{[1,0]}\,s^{-1}_{5}k^{-1}_{6\_3}$
$BK_{[3,0]} = k_{6\_3}\,K_{[3,0]}\,\beta_{[2,0]}\,K^{-1}_{[3,0]}k^{-1}_{6\_3}$
[3,0]
[3,1]
M4
$BK_{[4,1]} = k_{6\_3}s^{*}_{4}\beta_{[2,0]}\,s^{-1*}_{4}\,k^{-1}_{6\_3}$
M2
[4,0]
[4,1]
M3
$BK_{[4,0]} = k_{6\_3}\,s_{2}\beta_{[3,0]}\,s^{-1}_{2}\,k^{-1}_{6\_3}$
$BK_{[4,1]} = k_{6\_3}s_{3}\beta_{[3,0]}\,s^{-1}_{3}k^{-1}_{6\_3}$

Figure 4.9 Merge protocol : Authenticated key tree that M_3 received from merging group director, M_6



$K_{[0,0]} = K_{[1,0]}\, s_6^{*}\, \beta_{[0,0]}\, s_6^{-1*}\, K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_2\beta_3\beta_4\beta_5\beta_6$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\, \beta_{[0,0]}\, K_{[1,0]}^{-1}$
$K_{[1,0]} = K_{[2,0]}\, s_5 \beta_{[1,0]}\, s_5^{-1}\, K_{[2,0]}^{-1}$
$\beta_{[1,0]} = \beta_2\beta_3\beta_4\beta_5$
[1,0]
director
[1,1]
M_6
$BK_{[1,1]} = s_6^{*}\beta_{[0,0]}\, s_6^{-1*}$
$BK_{[2,0]} = K_{[2,0]}\, \beta_{[1,0]}\, K_{[2,0]}^{-1}$
$K_{[2,0]} = K_{[3,0]}\, s_4^{*}\, \beta_{[2,0]}\, s_4^{-1*}\, K_{[3,0]}^{-1}$
$\beta_{[2,0]} = \beta_2\beta_3\beta_4$
[2,0]
[2,1]
M_5
$BK_{[2,1]} = s_5\beta_{[1,0]}\, s_5^{-1}$
$BK_{[3,0]} = K_{[3,0]}\, \beta_{[2,0]}\, K_{[3,0]}^{-1}$
$K_{[3,0]} = s_2 s_3 \beta_{[3,0]}\, s_3^{-1} s_2^{-1}$
$\beta_{[3,0]} = \beta_2\beta_3$
[3,0]
[3,1]
M_4
$BK_{[4,1]} = s_4^{*}\beta_{[2,0]}\, s_4^{-1*}$
M_2
[4,0]
[4,1]
M_3
$BK_{[4,0]} = s_2\beta_{[3,0]}\, s_2^{-1}$
$K_{[4,0]} = s_2$
$BK_{[4,1]} = s_3\beta_{[3,0]}\, s_3^{-1}$

Figure 4.10 Merge Protocol : After M_2 computes keys and blinded keys

4.2.5 Partition Protocol

This operation is similar as the TBG protocol. The partition operation can occur when a network faults. The partition protocol actually presents a concurrent multiple members leaving from group. When multiple members *p* need to leave the group, the director is node above the undermost removing nodes in existing key tree. Otherwise, if the leaving node is child of root and the undermost removing nodes does not exist, the director is leaf node below the undermost the removing node. The same as the leave protocol, after the director deletes all leaving members from key tree, it selects new session random key, computes keys and blinded keys going up to the root and unicasts the key tree with authenticated blinded keys to reminder members. Finally, each member computes blinded keys and new group key. Figure 4.11 shows the initial situation before members leave from the group. Later, the example shows partition operation when all members in G_1 see M_2 and M_5 as leaving nodes and M_3 is director of G_1 because it is above the undermost removing node, M_2, before partition. While all members in G_2 see M_3, M_4, and M_6 as leaving members and M_5 is director of G_2 because it is above the undermost removing node, M_3, before partition. After partition process to two groups, Figure 4.12 shows authenticated key tree that M_4 received from director, M_3 in G_1 and Figure 4.13 shows authenticated key tree that M_2 received from director, M_5 in G_2. Figure 4.14 shows after M_3, M_2 computes keys and blinded keys in G_1 and G_2, respectively.

Then total communication message in partition protocol is one round that director unicasts message to reminding group members. Then the total amount of unicast message is *n-p* including the messages that director unicasts information to remaining group members except leaving members.

The conclusion of partition protocol is shown as follows:

Step 1: The director selects the new session random key, updates the key tree, computes keys and blinded keys and unicasts the new authenticated key tree.

$$M_d \xrightarrow{k_{d\_i} T^*[BK]\, k_{d\_i}^{-1}} \{\, M_i \,,\, i \in [1, n\text{-}p] \,\}$$

where $k_{d\_i} = x_d P_i x_d^{-1}$, x_d is director's long term private key and P_i is M_i's long term public key.

Step 2: The remaining members compute authenticated blinded key as $k_{i\_d}^{-1} k_{d\_i} T^*[BK]\, k_{d\_i}^{-1} k_{i\_d}$ to achieve the blinded keys in new key tree, $T^*[BK]$, where $k_{i\_d} = x_i P_d x_i^{-1}$.

Step 3: Each member computes the secret group key

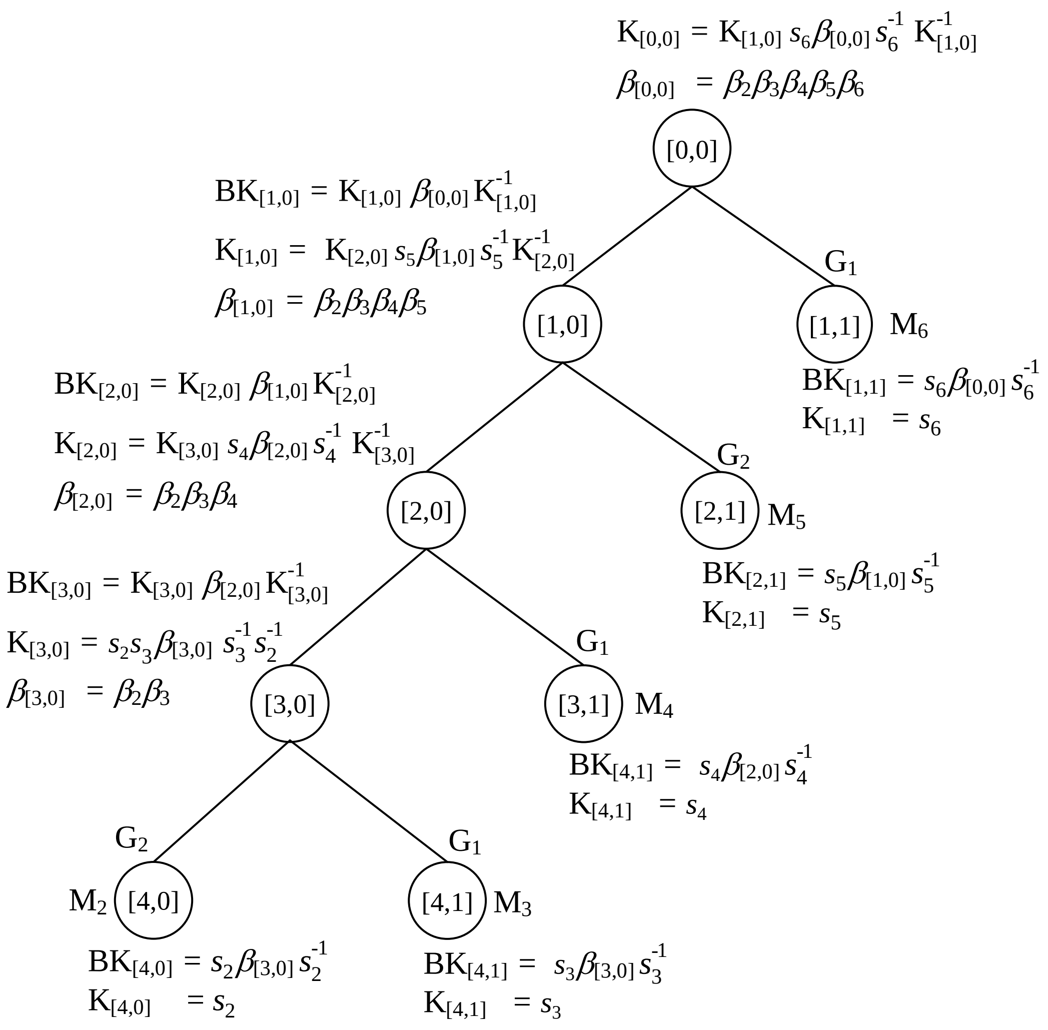

$K_{[0,0]} = K_{[1,0]}\, s_6 \beta_{[0,0]}\, s_6^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_2\beta_3\beta_4\beta_5\beta_6$
[0,0]
$BK_{[1,0]} = K_{[1,0]}\, \beta_{[0,0]} K_{[1,0]}^{-1}$
$K_{[1,0]} = K_{[2,0]}\, s_5 \beta_{[1,0]}\, s_5^{-1} K_{[2,0]}^{-1}$
$\beta_{[1,0]} = \beta_2\beta_3\beta_4\beta_5$
G_1
[1,0]
[1,1]
M_6
$BK_{[1,1]} = s_6 \beta_{[0,0]}\, s_6^{-1}$
$K_{[1,1]} = s_6$
$BK_{[2,0]} = K_{[2,0]}\, \beta_{[1,0]} K_{[2,0]}^{-1}$
$K_{[2,0]} = K_{[3,0]}\, s_4 \beta_{[2,0]}\, s_4^{-1} K_{[3,0]}^{-1}$
$\beta_{[2,0]} = \beta_2\beta_3\beta_4$
G_2
[2,0]
[2,1]
M_5
$BK_{[2,1]} = s_5 \beta_{[1,0]}\, s_5^{-1}$
$K_{[2,1]} = s_5$
$BK_{[3,0]} = K_{[3,0]}\, \beta_{[2,0]} K_{[3,0]}^{-1}$
$K_{[3,0]} = s_2 s_3 \beta_{[3,0]}\, s_3^{-1} s_2^{-1}$
$\beta_{[3,0]} = \beta_2\beta_3$
G_1
[3,0]
[3,1]
M_4
$BK_{[4,1]} = s_4 \beta_{[2,0]}\, s_4^{-1}$
$K_{[4,1]} = s_4$
G_2
G_1
M_2
[4,0]
[4,1]
M_3
$BK_{[4,0]} = s_2 \beta_{[3,0]}\, s_2^{-1}$
$K_{[4,0]} = s_2$
$BK_{[4,1]} = s_3 \beta_{[3,0]}\, s_3^{-1}$
$K_{[4,1]} = s_3$

Figure 4.11 Before tree updated: Partition Protocol

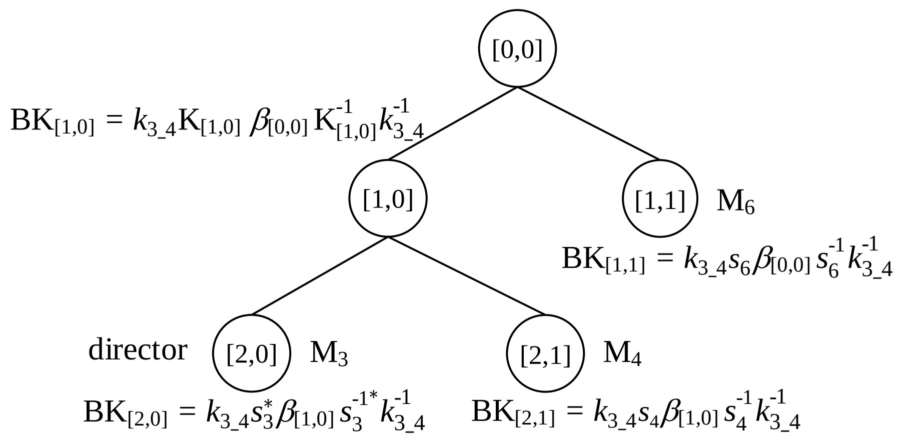

[0,0]
$BK_{[1,0]} = k_{3\_4} K_{[1,0]}\, \beta_{[0,0]} K_{[1,0]}^{-1} k_{3\_4}^{-1}$
[1,0]
[1,1]
M_6
$BK_{[1,1]} = k_{3\_4} s_6 \beta_{[0,0]}\, s_6^{-1} k_{3\_4}^{-1}$
director
[2,0]
M_3
[2,1]
M_4
$BK_{[2,0]} = k_{3\_4} s_3^{*} \beta_{[1,0]}\, s_3^{-1*} k_{3\_4}^{-1}$
$BK_{[2,1]} = k_{3\_4} s_4 \beta_{[1,0]}\, s_4^{-1} k_{3\_4}^{-1}$

Figure 4.12 Partition protocol : Authenticated key tree that M_4 received from director, M_3 in G_1

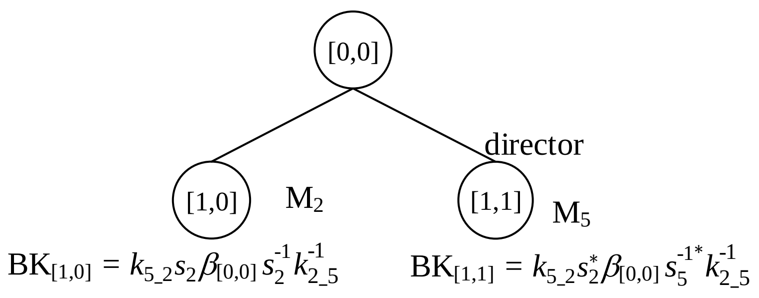

[0,0]
director
[1,0]
M$_2$
[1,1]
M$_5$
$BK_{[1,0]} = k_{5\_2} s_2 \beta_{[0,0]} s_2^{-1} k_{2\_5}^{-1}$
$BK_{[1,1]} = k_{5\_2} s_2^{*} \beta_{[0,0]} s_5^{-1*} k_{2\_5}^{-1}$

Figure 4.13 Partition protocol : Authenticated key tree that M$_2$ received from director, M$_5$ in G$_2$

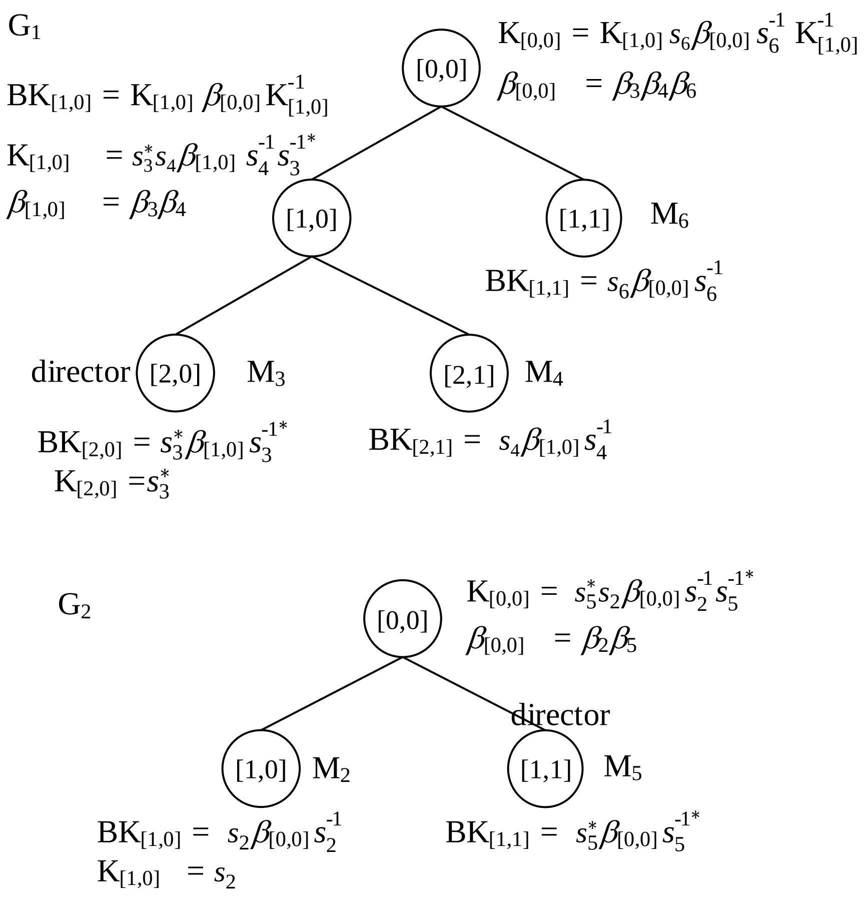

G$_1$
$K_{[0,0]} = K_{[1,0]} s_6 \beta_{[0,0]} s_6^{-1} K_{[1,0]}^{-1}$
$\beta_{[0,0]} = \beta_3 \beta_4 \beta_6$
[0,0]
$BK_{[1,0]} = K_{[1,0]} \beta_{[0,0]} K_{[1,0]}^{-1}$
$K_{[1,0]} = s_3^{*} s_4 \beta_{[1,0]} s_4^{-1} s_3^{-1*}$
$\beta_{[1,0]} = \beta_3 \beta_4$
[1,0]
[1,1]
M$_6$
$BK_{[1,1]} = s_6 \beta_{[0,0]} s_6^{-1}$
director
[2,0]
M$_3$
[2,1]
M$_4$
$BK_{[2,0]} = s_3^{*} \beta_{[1,0]} s_3^{-1*}$
$K_{[2,0]} = s_3^{*}$
$BK_{[2,1]} = s_4 \beta_{[1,0]} s_4^{-1}$
G$_2$
[0,0]
$K_{[0,0]} = s_5^{*} s_2 \beta_{[0,0]} s_2^{-1} s_5^{-1*}$
$\beta_{[0,0]} = \beta_2 \beta_5$
director
[1,0]
M$_2$
[1,1]
M$_5$
$BK_{[1,0]} = s_2 \beta_{[0,0]} s_2^{-1}$
$K_{[1,0]} = s_2$
$BK_{[1,1]} = s_5^{*} \beta_{[0,0]} s_5^{-1*}$

Figure 4.14 Partition Protocol : After M$_3$, M$_2$ computes keys and blinded keys in G$_1$ and G$_2$, respectively

4.2.6 Key Refreshing

Key refreshing in MANETs is necessary, since most nodes can be easily compromised due to their mobility and physical vulnerability. Then the key refreshing should be occurred periodically in order to limit exposure due the loss of keys and limit the amount of ciphertext available to cryptanalysis for given group key. In this protocol the node that needs to refresh the key acts as the director. In similar way of other protocols, the director chooses the new session random key, computes keys and blinded keys up to the root, and unicast updated key tree with authenticated blinded keys. All members compute blinded keys and new group key. The conclusion of key refreshing protocol is shown as follows:

Step 1: The director (refreshing node) selects new session random key, computes keys and blinded keys and unicasts new key tree containing authenticated blinded key.

$$\mathrm{M}_d \xrightarrow{\;k_{d\_i}\mathrm{T}^*[\mathrm{BK}]\,k_{d\_i}^{-1}\;} \{\,\mathrm{M}_i\,,\ i \in [1, n]\,\}$$

where $k_{d\_i} = x_d\, \mathrm{P}_i\, x_d^{-1}$, x_d is director's long term private key and P_i is M_i's long term public key.

Step 2: Each member computes authenticated blinded key as $k_{i\_d}^{-1}\, k_{d\_i}\, \mathrm{T}^*[\mathrm{BK}]\, k_{d\_i}^{-1}\, k_{i\_d}$ to achieve the blinded keys in new key tree, $\mathrm{T}^*[\mathrm{BK}]$, where $k_{i\_d} = x_i\, \mathrm{P}_d\, x_i^{-1}$.

Step 3: Each member computes the secret group key

4.3 Security Analysis

As described above, it can see that Group Key Agreement on Tree-based Braid Groups satisfies forward and backward secrecy. It also satisfies key independence. Moreover, it shares the group key without the man-in-the-middle attack by using authentication scheme. The passive adversaries are unable to compute future and previous group key although they know all previous key trees and new key tree respectively, since the director refreshes the session random key every event.

First, the protocol is considered the forward secrecy, note that members that leave the group or passive adversaries who know a contiguous subset of old group key are unable to compute future group key. The forward secrecy is determined in leave and partition event. Assume *A* as leaving member at position *a* in key tree *T*. *A* knows all secret keys on key-path that are valid during its group membership. However the director of the leave and partition event updates own session random key and causes the change of keys and blinded keys. Therefore *A* is unable to compute the subsequent group key, because the key tree information is changed. Thus the protocol provides the forward secrecy.

Later, the protocol is considered the backward secrecy to show that new group members are unable to compute old group keys. Assume *A* becomes a new member at position *a* in key tree *T*. As a new member *A* is able to compute all keys on key-path. The director of the join and merge event updates own session random key and causes the change of keys and blinded keys in key-path. Therefore *A* is unable to compute previously used group key, since *A* can only compute new group keys due to changed key tree information. Therefore the protocol satisfies the backward secrecy.

The combination of forward and backward secrecy, can conclude that ATBG protocol satisfies key independence.

Finally, the research is considered the man-in-the-middle attack to show that intruder cannot impersonate each member to the satisfaction of the other in any membership event. Let C who can modify, delay or inject messages, is an active opponent. The objective of opponent is sharing a key with either member by disguising as some member. The attack on some member discusses as follows: C intercepts the message to eavesdrop and possible deliver a false message to Alice and Bob. First, Alice asks Bob for his blinded key. If Bob send his blinded key to Alice, a man-in-the-middle-attack can begin if C is able to intercept it. C send message to Alice that claims to be from Bob, but instead includes C’s blinded key. Alice sends blinded key that to Bob with authenticated version that is $k_a BK_a k_a^{-1}$ where $k_a = x_a P_b x_a^{-1}$, x_a is Alice's long term private key and P_b is Bob's long term public key. The Alice's message is intercepted by C and forged message. The forged message is sent to Bob. Meanwhile, Bob sends $k_b BK_b k_b^{-1}$ to Alice, and is also intercepted by C. C forges message and sends it to Alice. The computing $k_a^{-1} k_b BK_b k_b^{-1} k_a$ and $k_b^{-1} k_a BK_a k_a^{-1} k_b$ are difficult without the knowledge of k_a and k_b respectively. Therefore C difficult compute blinded key of Alice and Bob.

4.4 Complexity Analysis

4.4.1 Communication Cost

The communication cost is shown in Table 4.1 that compared among TBG, and ATBG. The number of rounds on both protocols is constant and equal in all operations. Therefore this study can conclude that the amount of protocol rounds do not depend on the number of members. The amount of messages that is sent among group members, are constant except setup protocol. The amount of messages in setup protocol depends on the number of group members. The total number of message in setup, join and merge protocol of ATBG is more than the previous protocol TBG due to the authentication process, otherwise it is same. The number of unicast messages in merge protocol depends on the amount of merging group member. The ATBG is efficient lower than TBG but it stronger.

4.4.2 Computation Cost

The computation cost in Table 4.2, the computation cost of both protocols including TBG and ATBG are same, because the ATBG protocol is extended version with authentication of TBG.

Table 4.1 Communication Cost of ATBG Protocol

Protocol	Operation	Rounds	Message	Unicast Message	Multicast Message
TBG	Setup	2	*n*	*n*-1	1
	Join	2	2	1	1
	Leave	1	1	0	1
	Merge	2	2	1	1
	Partition	1	1	0	1
ATBG	Setup	*N*	2(*n*-1)	2(*n*-1)	0
	Join	2	*n*+1	*n*+1	0
	Leave	1	*n*-1	*n*-1	0
	Merge	2	*n+m*	*n+m*	0
	Partition	1	*n-p*	*n-p*	0

Table 4.2 Computation Cost of ATBG Protocol

Protocol	Operation	Permutation
TBG	Setup	*N*
	Join	2
	Leave	*n*-1
	Merge	*n+m*
	Partition	*n-p*
ATBG	Setup	*N*
	Join	2
	Leave	*n*-1
	Merge	*n+m*
	Partition	*n-p*

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

The research proposes tree-based group key agreement on braid groups and extends to authenticated version. The modified STR using braid groups instead of Diffie-Hellman supports dynamic membership group operation including join, leave, merge and partition with satisfied forward and backward secrecy. My both protocols involve braid groups operation including product and inverse with key tree whose computation cost is much lower than modular exponentiation in STR and braid groups on GDH. My protocols are fully contributory scenario for key agreement that not require the trust party or long-term controller in online operation to avoid the problems with the centralized trust and the single point of failure. My protocols avoid the member serialization by using key tree. A number of existing protocols require group member sequencing that in mobile ad hoc networks is not efficient since the sequence may not correspond to the best geographic node placement and may lead to increase communication cost. Therefore communication cost in my protocols is less than braid groups on GDH protocol. Moreover protocols reduce the computation cost in group event while preserving the constant round communication and the security property. Finally ATBG protects man-in-the-middle-attack by using identity authentication. Therefore TBG and ATBG are suitable for environment of mobile ad hoc networks.

The limitation of my protocol is discussed as follows. My protocols do not consider the key length that using for each member. The key length effect the performance in computation cost and security strength. The shorter key length leads to weaker security strength but faster computation. Otherwise the longer key length leads to stronger security strength but slower computation.

Moreover, in merge protocol, the merging group director in previous operation before merging process is point of failure in the protocol. The merging group director has to exist before needing to merge group. The authenticated group key agreement protocol does not consider in authenticated phase. If the authentication of members fails because the member needing to join the group is not group member, how the protocol can handle in situation?

5.2 FUTURE WORKS

In the future works, the protocol should be adapted to completely solution. First In mobile ad-hoc networks possible occurs multi-hop communication lead to more overhead in communication between members. The protocol should be considered about communication cost of non-members in all operations. Later the total simulation should be done in any variable such as key length, number of members, area etc. The result of simulation can be considered to adapt the protocol.

BIBLIOGRAPHY

Abdel-Hafex, A.; Miri, A. and Orozco-Barbosa, L. 2004. Authenticated Secure Communications in Wireless Networks. **The Fifth European Wireless Conference: Mobile and Wireless Systems beyond 3G.** Barcelona, Spain: Technical University of Catalonia.

Anshel, I.; Anshel, M. and Fisher, B. 2001. New Key Agreement Protocols in Braid Group Cryptography. In **Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA.** London: Springer-Verlag. Pp. 13-27.

Anton, E. and Duarte, O. 2002. Group Key Establishment in Wireless Ad Hoc Networks. In **Workshop on Quality of Service and Mobility**. Angra dos Reis, RJ.

Anzai, J. and Matsumoto, T. 2005. A Distributed User Revocation Scheme for Ad-Hoc Networks**.** In **IEICE Transactions and Communications.** E88-B (September): 3623-3634.

Asokan, N. and Ginzboorg, P. 1999. Key Agreement in Ad-hoc Networks. **Computer Communications**. 23 (Feb): 1627-1637.

Ateniese, G.; Steiner, M. and Tsudik, G. 2000. New Multiparty Authentication Services and Key Agreement Protocols**. IEEE Journal of Selected Areas in Communications**. 18 (April): 628-639.

Augot, D.; Bhaskar, R.; Lssarny, V. and Sacchetti, D. 2005. An Efficient Group Key Agreement Protocol for Ad hoc Networks. In **6th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks.** Washington, DC: IEEE Computer Society. Pp. 576-580.

Balachandran, R. K.; Ramamurthy, B.; Zou, X. and Vinodchandran, N.V. 2005. CRTDH: An Efficient Key Agreement Scheme for Secure Group Communications in Wireless Ad Hoc Networks. In **IEEE International Conference on Communications.** Pp. 1123-1127.

Basagni, S.; Herrin, K.; Rosti, E. and Bruschi, D. 2001. Secure Pebblenets. In **ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)**. New York: ACM. Pp. 156-163.

Becker, K. and Wille, U. 1998. Communication Complexity of Group Key Distribution. In **Proceedings of the 5th ACM conference on Computer and communications security**. New York: ACM. Pp. 1-6.

Birman, J.; Ko, K. H. and Lee, S. J. 1998. A New Solution to the Word and Conjugacy Problems in the Braid Groups. **Advances in Mathematics**. 139: 322-353.

Bresson, E.; Chevassut, O.; Essiari, A. and Pointcheval, D. 2003. Mutual Authentication and Group Key Agreement for Low-Power Mobile Device. **Proceeding of the 5th IFIP-TC6 International Conference on Mobile and Wireless Communication Networks**. Singapore: IFIP. Pp. 59-62.

Burmester, M. and Desmedt, Y. 1994. A Secure and Efficient Conference Key Distribution System. **Advances in Cryptology (EUROCRYPT'94).** Perugia, Italy: Springer. LNCS 950: 275-286.

Cha, J. C.; Ko, K. H.; Lee, S. J.; Han, J. W. and Cheon, J. H. 2001. An Efficient Implementation of Braid Groups. **ASIACRYPT 2001**. Queensland, Australia: IACR. LNCS 2248: 144-156.

Chaturvedi, A. and Lal, S. 2008. An Authenticated Key Agreement Protocol Using Conjugacy Problem in Braid Groups. **International Journal of Network Security**. 6 (March): 181–184.

Hubaux, J. P.; Buttyan, L.; and Capkun, S. 2001. The Quest for Security in Mobile Ad Hoc Networks. In **ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)**. New York: ACM. Pp. 146-155.

Inoue, D. and Kuroda, M. 2004. FDLKH: Fully Decentralized Key Management Scheme on Logical Key Hierarchy. In **American Conference on Neutron Scattering.** Berlin / Heidelberg: Springer. 3089:339-354

Kim, Y.; Perrig, A. and Tsudik, G. 2000. Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups. In **Proceedings of the 7th ACM Conference on Computer and Communications Security**. New York: ACM. Pp. 235-244.

Kim, Y.; Perrig, A. and Tsudik, G. 2001. Communication-Efficient Group Key Agreement. In **Information Systems Security, Proceedings of the 17th International Information Security Conference IFIP SEC'01**. Netherlands: Kluwer. Pp. 229-244.

Kim, Y.; Perrig, A. and Tsudik, G. 2004. Tree-based Group Key Agreement. **ACM Transactions on Information and System Security**. 7 (1): 60-96.

Ko, K. H.; Lee, S. J.; Cheon, J. H.; Han, J. W.; Kang, J. and Park, C. 2000. New Public-Key Cryptosystem Using Braid Groups. **Proceedings of Crypto 2000.** Santa Barbara, California: Springer-Verlag. LNCS 1880: 166-183.

Kong, J.; Lee, Y. Z. and Gerla, M. 2006. Distributed Multicast Group Security Architecture for Mobile Ad Hoc Networks. In **Wireless Communications and Networking Conference.** Las Vegas, NV: IEEE**.** Pp. 640-645.

Kong, J.; Zerfos, P.; Luo, H.; Lu, S. and Zhang, L. 2001. Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks. In **IEEE International Conference on Network Protocols**. California: IEEE. Pp. 251-260.

Kui, R. and Gang, Y. 2004. Efficient Key Agreements in Ad-hoc Networks. In **Proceedings of 8th Conference of China Cryptography**. Shanghai, China: Science Press.

Law, L.; Menezes, A.; Qu, M.; Solinas, J. and Vanstone, S. 2003. An Efficient Protocol for Authenticated Key Agreement Protocol. **Design, Codes and Cryptography**. 28 (2): 119–134.

Lazos, L. and Poovendran, R.. 2005. Power Proximity Based Key Management for secure Multicast in Ad Hoc Networks. **Wireless Networks**. 13 (January): 127-148.

Lee, P.; Lui, J. and Yau, D. 2006. Distributed Collaborative Key Agreement Protocols for Dynamic Peer Groups. **IEEE/ACM Transactions on Networking**. 14 (April): 263-276.

Li, X.Y.; Wang, Y. and Frieder, O. 2002. Efficient Hybrid Key Agreement Protocol for Wireless Ad-Hoc Networks. In **Proceedings of 11th International Conference on Computer Communications and Networks**. Miami, Florida:IEEE. Pp. 404-409.

Maki, S.; Aura, T. and Hietalahti, M. 2000. Robust Membership Management for Ad-hoc Groups. In **Proceedings of the 5th Nordic Workshop on Secure IT Systems**. Reykjavík, Iceland: Reykjavík University.

Manulis, M. 2005. Key Agreement for Heterogeneous Mobile Ad-Hoc Groups. **Proceeding of the 11th International Conference on Parallel and Distributed Systems**. Fukuoka, Japan: IEEE. Pp. 290-294.

Nukherjee, A.; Gupta, A. and Agrawal, D. P. 2005. Totally Distributed Key Management for Dynamic Groups in MANETs. In **IEEE International Performance Computing, and Communications Conference.** Phoenix: Arizona: IEEE. Pp. 185-192.

Pereira, O. and Quisquater, J. 2001. A Security Analysis of the Cliques Protocols Suites. In **Proceedings of the 14th IEEE Computer Security Foundations Workshop**. Washington, DC: IEEE Computer Society. Pp. 73-81.

Ressson, E. and Catalano, D. 2004. Constant Round Authenticated Group Key Agreement from General Assumptions**.** In **Proceeding of Public Key Cryptography.** Singapore: Springer. Pp. 115-129.

Rhee, K. H.; Park, Y. H. and Tsudik, G. 2005. A Group Key Management Architecture for Mobile Ad-hoc Wireless Network**. Journal of Information Science and Engineering.** 21:415-428.

Seys, S. and Preneel, B. 2005. The Wandering Nodes: Key Management for Low-power Mobile Ad Hoc Network. In **Proceeding of the 25th IEEE International Conference on Distributed Computing System**. Columbus, Ohio: IEEE. Pp. 916-922.

Shpilrain, V. and Ushakov, A. 2008. An Authentication Scheme Based on the Twisted Conjugacy Problem. **Applied Cryptography and Network Security**. LNCS 5037: 366-372.

Sibert, H.; Dehornoy, P. and Girault, M. 2006. Entity Authentication Schemes Using Braid Word Reduction. **Discrete Applied Mathematics**. 154 (February): 420-436.

Steer, D.; Strawczynski, L.; Diffie, W. and Wiener, M. 1988. A Secure Audio Teleconference System. In **Advances in Cryptology - CRYPTO'88**. Santa Barbara, California: Springer. LNCS 403 (August): 520-528.

Steiner, M.; Tsudik, G. and Waidner, M. 1998. Cliques: A New Approach to Group Key Agreement. In **International Conference on Distributed Computing Systems**. Amsterdam, Netherlands: IEEE. Pp. 380-387.

Steiner, M.; Tsudik, G. and Waidner, M. 2000. Key Agreement in Dynamic Peer Groups. **IEEE TRANSACTIONS on Parallel and Distributed Systems.** 11 (August): 769-780.

Wang, L. and Wu, C.K. 2006. Efficient Key Agreement for Large and Dynamic Multicast Groups. **International Journal of Network Security**. 3 (July): 8-17.

Wallner, D.; Harder, E. and Agee, R. 1999. Key Management for Multicast: Issue and architecture. **RFC**. 2676.

Yao, G.; Ren, K.; Bao, F.; Deng, R. H. and Feng, D. 2003. Making the Key Agreement Protocol in Mobile Ad Hoc Network More Efficient. In **International Conference on Applied Cryptography and Network Security**. Kunming, China: Springer-Verlag. Pp. 343-356.

Yasinsac, A.; Thakur, V.; Carter, S. and Cubukcu, I. 2002. A Family of Protocols for Group Key Generation in Ad Hoc Networks. In **IASTED International Conference on Communication and Computer Networks**. Innsbruck, Austria: IASTED. Pp. 183-187.

Yi, S. and Kraverts, R. 2002. Key Management in Heterogeneous Ad Hoc Wireless Networks. In **IEEE International Conference on Network Protocols**. Paris, France: IEEE. Pp. 202-205.

Yu, W.; Sun, Y. and Liu, K. J. R. 2005. Minimization of Rekeying Cost for Contributory Group Communication. In **IEEE Global Telecommunications Conference.** St. Louis, Missouri: IEEE. Pp. 1716-1720.

Zhou, L. and Ravishankar, C. V. 2004. Efficient, Authenticated, and Fault-Tolerant Key Agreement for Dynamic Peer Groups. In **Third International IFIP-TC6 Networking Conference.** Athens, Greece: Springer. 3042: 759-770.

Zhu, S.; Xu, S.; Setia, S. and Jajodia, S. 2003. Establishing Pair-Wise Keys for Secure Communication in Ad-Hoc Networks: A Probabilistic Approach. In **IEEE International Conference on Network Protocols.** Atlanta, GA: IEEE.

BIOGRAPHY

Name	Thanongsak Aneksrup
ACADEMIC BACKGROUND	M.E. (Civil Engineering), Asian Institute of Technology, Thailand. B.E. (Civil Engineering). Royal Thai Air Force Academy, Thailand.
PRESENT POSITION	Engineer in Directorate of Civil Engineering, Royal Thai Air Force. Lecturer in Department of Computer Sciences, Siam University.
EXPERIENCES	Software Analyst in Core Bank System, Exim Bank. ICT Consultant in Department of Business Development.