University of Nebraska - Lincoln

DigitalCommons@University of Nebraska - Lincoln

CSE Conference and Workshop Papers

Computer Science and Engineering, Department of

2005

DGKD: Distributed Group Key Distribution with Authentication Capability

Pratima Adusumilli

Xukai Zou

Byrav Ramamurthy
University of Nebraska-Lincoln, bramamurthy2@unl.edu

Digital Object Identifier: 10.1109/IAW.2005.1495965

Proceedings of the 2005 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY

DGKD: Distributed Group Key Distribution with Authentication Capability

Pratima Adusumilli, Xukai Zou, Byrav Ramamurthy

Abstract— *Group key management (GKM) is the most important issue in secure group communication (SGC). The existing GKM protocols fall into three typical classes: centralized group key distribution (CGKD), decentralized group key management (DGKM), and distributed/contributory group key agreement (CGKA). Serious problems remains in these protocols, as they require existence of central trusted entities (such as group controller or subgroup controllers), relaying of messages (by subgroup controllers), or strict member synchronization (for multiple round stepwise key agreement), thus suffering from the single point of failure and attack, performance bottleneck, or misoperations in the situation of transmission delay or network failure. In this paper, we propose a new class of GKM protocols: distributed group key distribution (DGKD). The new DGKD protocol solves the above problems and surpasses the existing GKM protocols in terms of simplicity, efficiency, scalability, and robustness.*

Keywords: Secure Group Communication, Group Key Management, Centralized Key Distribution, (Distributed) Contributory Key Agreement, Distributed Key Distribution.

I. Introduction

Secure group communications (SGC) over networks (e.g, the Internet) refers to a setting in which a group of members can send messages to and receive messages from group members, in a way that outsiders are unable to glean any information even when they are able to intercept the messages. SGC is an inseparable component of cyber security. Broad critical applications such as collaborative work, teleconferencing/medicine, multi-partner military action, and cyber forensics in critical fields depend on SGC for their security.

The most important problem facing SGC is group key management (GKM). The primary difficulty for GKM comes from member dynamics. How to design robust, scalable, efficient GKM protocols supporting high dynamics is the focus of all SGC researches. Many GKM protocols have appeared in the literature and typically fall into three categories: centralized group key distribution (CGKD), decentralized group key management with relaying (DGKM), and (distributed) contributory group key agreement (CGKA).

In CGKD schemes [1], [2], [3], [4], [5], [6], [7], [8], [9], there is a central trusted authority (called group controller (GC)) that is responsible for generating and distributing the group key. Whenever a new member joins or an existing member leaves, the GC generates a new group key and distributes the new key to the group. The problems with the centralized schemes are the central point of failure, performance bottleneck, non-scalability, and the requirement of trustworthiness of the group controller by all members. In DGKM schemes [10], [11], [12], [13], the group is divided into multiple distinct subgroups and every subgroup has a subgroup controller (SC) responsible for key management for its subgroup. In addition, an SC has the key of its parental subgroup. When an SC receives a message from one subgroup, it decrypts the message, encrypts the message with the key of the other subgroup and sends to the other subgroup, i.e., relaying the message. The problems with DGKM are that SCs can still be considered as central and trusted entities (at a smaller scale) and the messages undergo multiple relaying before they reach the entire group. Relaying of every data message puts huge burden on SCs. In CGKA schemes [14], [14], [15], [16], [17], [18], the group key is generated/agreed up by uniform contributions from all group members. These kind of schemes assume equality and uniform work load among group members. They are generally executed in multiple rounds and require strict synchronization. The CGKA protocols are primarily different variations of the n-party Diffie-Hellman key agreement/exchange [14], [16], [19], [20], [17], [18]. The main problem with using this key exchange mechanism is that the group members need synchronization to iteratively form parental keys from their two children's keys. Once one member is slow or one rekeying packet is delayed, the key agreement process will be postponed or even misoperates. Moreover, there are dependances among nodes' keys (i.e., a blinded node key is dependent on the secret node key and a parental key on its two child's keys). This dependance results in the breaking of all ancestral keys once one key is compromised.

To overcome the above problems we propose a new class of GKM protocols: called distributed group key distribution (DGKD). The DGKD protocol does not assume any trusted and more powerful third party but allows the equality of capability, responsibility, and trustiness among all group members. The protocol organizes the members in a tree structure and performs any rekeying operation in just two rounds, which do not need to be strictly synchronized.

This work was partially supported by the U.S. NSF grant CCR-0311577.

P. Adusumili and X. Zou: School of Science, Purdue University at Indianapolis, Indianapolis, IN.

B. Ramamurthy: University of Nebraska-Lincoln, Lincoln, NE.

0-7803-9290-6/05/$20.00 ©2005 IEEE.

The new protocol also allows strong yet simple authentication. In addition, DGKD has the following advantages: (1) one key (not two keys) per node; (2) independance of nodes' keys; (3) robust against transmission delay, network failure or compromise of node keys. All these properties make the new protocol simple, robust, efficient and scalable.

The rest of the paper is organized as follows. Section II briefly describes the related work in the area of SGC. We propose the new protocol in Section III and the issues of performance and security are discussed in Section IV. Finally we conclude the paper in Section V.

II. Related work

Extensive research has been conducted on GKM and a considerable number of protocols have been developed [21], [22], [23], [24], [25], [26], [27], [28], [29], [14], [3], [30], [31], [15], [10], [32], [33], [4], [5], [16], [34], [35], [36], [37], [38], [12], [7], [39], [40], [41], [17], [18], [42], [43], [44], [9], [45], each with different properties and performance.

SGC applications can typically be divided into *broadcast/multicast* communication, i.e., one sender and multiple receivers, or *one-to-many* communication and *group* (or *many-to-many*) communication, i.e., every sender also being a receiver. Some GKM schemes [21], [46], [15], [10], [11], [13] are suitable for broadcast applications, some other schemes [28], [14], [15], [16], [41], [17], [18] for many-to-many applications, and there are also some schemes [3], [12], [7], [9] suitable for both kinds of applications. Based on how the *group key* is formed and distributed, the GKM protocols are classified as CGKD, DGKM, and CGKA. Based on the *kind of cryptosystem* used, the schemes for SGC can be divided into *public-key based schemes* [46], [11], [13] and *secret-key based schemes*. Based on the *kind of security*, the SGC schemes may be classified as *unconditionally secure* or *computationally secure* [47], [43]. Furthermore some schemes may resist against any number of colluding adversaries, whereas others [22], [23], [24], [25], [26], [27], [32], [43] only resist against the collusion of up to certain number of adversaries. For a comprehensive survey of state-of-art techniques and challenging problems in the area of SGC, readers are referred to the book "secure group communications over data networks", which is published by Springer [48].

Among all the GKM protocols, the tree based GKM scheme (with various variants) [1], [2], [3], [49], [19], [6], [7], [41], [8], [9], [50], [51] is the most typical approach. The scheme is simple, efficient, scalable, and easy to implement. The scheme can be used for both one-to-many multicast communication as well as many-to-many group communication. Moreover the tree based GKM scheme has versions of both CGKD and CGKA.

III. Distributed Group Key Distribution (DGKD): a new class of GKM Protocols

A. Principle and assumption

There are some assumptions in existing schemes. In CGKD/DGKM, a secure channel is assumed to exist between the GC/SC and each of the potential group members/subgroup members. This secure channel is generally implemented by public key cryptosystems. In CGKA, which is typically based Diffie-Hellman key exchange which suffers from the Man-in-the-Middle attack, it is assumed that each group member is equipped with some authentication capability which is also implemented by public key cryptosystems. Similarly, DGKD assumes that every group member has a publicly known (unforgeable) public key.

The new DGKD protocol adopts a tree structure and utilizes three basic mechanisms to implement distributed key generation and distribution: 1) the leaf key of a node is the public key of the corresponding group member and all the intermediate nodes' keys are secret keys, 2) the sponsor of a joining or leaving member initiates the key generation and rekeying process and sends the new keys to co-distributors (i.e., the first round), 3) the co-distributors then help distribute the new keys to group members in a distributed/parallel manner (i.e., the second round).

All group members have the same capability and are equally trusted. Also, they have equal responsibility, i.e. any group member could be a potential sponsor of other members or a co-distributor (depending on the relative locations of the member and the joining/leaving members in the tree). Thus there is no dependance on a single entity and even if a sponsor node fails a new sponsor for the joining/leaving member is chosen by other members. This improves the robustness of the protocol.

B. Sponsor

A sponsor is a member and the sponsor of a subtree is defined as the member hosted on the rightmost leaf in the subtree (note: "rightmost" can be equally replaced with "leftmost"). Every node has an associated sponsor field as shown in Figure 1.

The sponsor field at a particular node is updated when it is along the joining or leaving member's path. We show the joining algorithm for updating the sponsor field in Figures 2.

When a member joins, the sponsor field along the joining members path is updated from bottom to the root. If the new members id is greater than the sponsor id of the node then update the sponsor id with the new member's id. This is continued until the root (See Figure 3).

When m_7 joins, the sponsor field along its path is updated. The sponsor id of the node k_{6-7} is lesser than the id of m_7, so it is updated to 111. Similarly the sponsor id's of nodes k_{4-7} and k_{0-7} are updated to 111. Whenever the

0-7803-9290-6/05/$20.00 ©2005 IEEE.

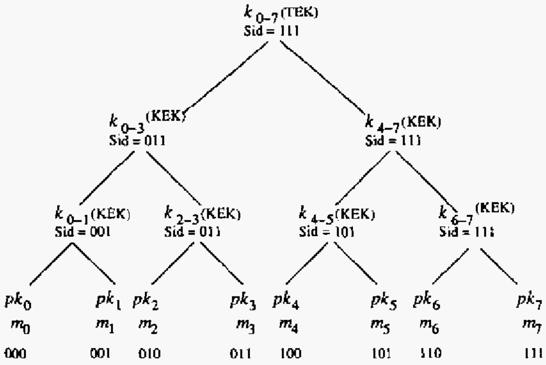

Fig. 1. A tree showing sponsor for each node.

```
Every member
        .iterate over all the nodes along the joining
        members path from leaf to the root
                .if the joining members id is greater than the
                sponsor id for that node
                        .sponsor id = joining members id
                        .continue
                . else
                        .break
```

Fig. 2. Sponsor update: Join.

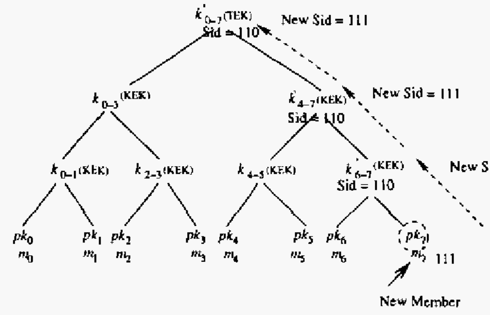

Fig. 3. Updating the sponsor field when a member joins.

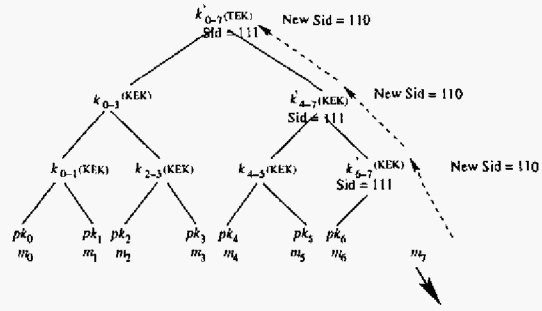

Fig. 4. Updating a sponsor field when a member leaves.

sponsor id for a node is greater than the joining members id then the check can be stopped.

When a member leaves, every member checks along the path of the leaving member to update the sponsor field. If a node has the leaving member as the sponsor then they update the sponsor field with the sponsor id/member id of the other child if exists. This continues upto the root (See Figure 4).

When m_7 leaves, the sponsor field along its path is updated. Since the leaving member is the sponsor all along its path, the sponsor field has to be updated by checking for the new sponsor for all the nodes. m_6 becomes the new sponsor for node k_{6-7}. For node k_{4-7} the member ids of both its children are compared and the greater becomes the new sponsor, in this case m_6. This continues until the root.

C. Co-distributors

When a sponsor changes the keys along the path, it needs to distribute them. The sponsor has to distribute the keys to all the members whose keys have been changed. But it does not know the keys along the other paths to distribute the new keys. So, a co-distributor is required to distribute them. The co-distributor is the sponsor of a node on another path whose key is not known to the original sponsor. The sponsor encrypts the changed key with the co-distributors public key and broadcasts this information. Thus, the co-distributor helps the sponsor in distributing the changed common keys along the other paths.

D. Initial group key generation and distribution Protocol

Suppose n members m_1,......,m_n decide to form a group. They build a virtual key tree and selects a sponsor to decide an order in which they join the tree. Every member updates the key tree by adding members in the key tree based on that order and they update the sponsor field in all the intermediate nodes. Then every member checks if it is responsible for generating any keys along its path. If so, it generates them and distributes the keys either directly or with the help of co-distributors. When two sponsors are responsible for generating the same key then the rightmost among them generates it. As more members join the key tree the sponsors and the height of the key tree increase.

As illustrated in Figure 5, m_7, m_5, m_3 and m_1 are responsible for generating the keys. m_7 generates all the keys (k_{6-7}, k_{4-7} and k_{0-7}) along its path to the root. Then it encrypts as follows and broadcasts: $\{k_{6-7}, k_{4-7}, k_{0-7}\}_{pk_6}$, and $\{k_{0-7}, k_{4-7}\}_{pk_5}$. m_5 will decrypt k_{0-7} and k_{4-7} and encrypt it as $\{k_{0-7}, k_{4-7}\}_{k_{4-5}}$ where k_{4-5} is generated by m_5 and sent to m_4. Similarly keys are generated by m_3 in the left subtree along its path and the root key which is generated by the rightmost sponsor m_7 is sent to the co-distributor of the left subtree m_3 as follows. $\{k_{0-7}\}_{pk_3}$

0-7803-9290-6/05/$20.00 ©2005 IEEE.

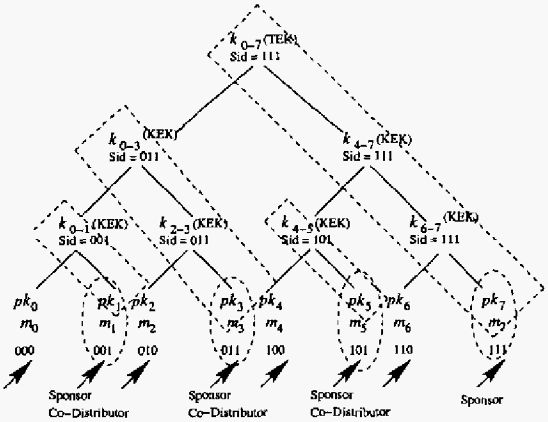

Fig. 5. Initial key generation Example

is broadcast and m_3 will decrypt k_{0-7} and encrypts it as $\{k_{0-7}\}_{k_{0-3}}$ and broadcasts it. Thus every member has the newly generated keys along its path. Only two rounds are required for this protocol, one round for generating keys and distributing along the path and another for co-distributors to distribute them.

E. Join protocol

```
Step 1: New member broadcasts request for join
   . $m_{n+1}$ $(PK_{n+1})$ ———> $m_1,..,m_n$
Step 2: Every member
      . updates the key tree by adding a new member node
      . Find sponsor for joining member:
            . if sibling present, sponsor = sibling
            . else sponsor = $m_{n+1}$
      . update the sponsor field along the path of the joining
      member to the root if required
Step 3: If joining member's sponsor is itself
      . generates new secret keys along the joining members path
      and distribute them to co-distributors and to other members
      directly by encrypting with common key and broadcasting
Step 4: If co-distributor is itself
      . encrypt the key sent by the joining members sponsor with
      appropriate key and broadcast
```

Fig. 6. Join Protocol.

Suppose there are n members in the group $m_1,......,m_n$. A new member m_{n+1} makes a join request by broadcasting its public key PK. The rightmost member in the key tree authenticates the new member, decides the insertion location for the new member and broadcasts this information to other members. Additionally the rightmost member also sends the virtual key tree and list of public keys of other members to the new member. All other members update the key tree by adding a new member node in the specified location. Then every member checks to see if it is the sponsor of the joining member. If the new member has a sibling it becomes the sponsor and generates new keys along the path.- If there is no sibling then the joining member itself becomes the sponsor and generates the new keys along its path and distributes them. Members update the sponsor field appropriately if required. Figure 6 describes the join protocol and Figure 7 shows the protocol operation when a new member joins.

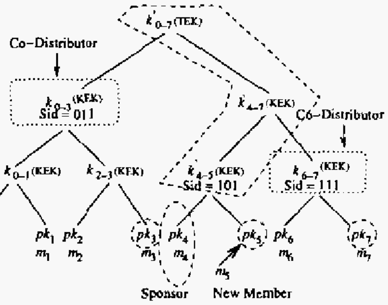

Fig. 7. A new member joins (becomes m_5), m_4 is sponsor and m_3 and m_7 are co-distributors.

When a new member joins, m_7 determines the position(i.e., m_5) and places the member there. m_7 broadcasts the position of the new member to other members. All members also determine that m_4 is the sponsor of m_5. So m_4 initiates the rekeying process as follows: 1) generates new keys k'_{4-5}, k'_{4-7}, and k'_{0-7}. 2) after determining the co-distributors m_3 and m_7, encrypts as follows and broadcasts: $\{k'_{4-7}, k'_{0-7}\}_{pk_7}$, and $\{k'_{0-7}\}_{pk_3}$, 3). m_3 will decrypt k'_{0-7} and encrypt it as $\{k'_{0-7}\}_{k_{0-3}}$ and m_7 will decrypt k'_{4-7} and k'_{0-7} and encrypt them as $\{k'_{4-7}\}_{k_{6-7}}$ and $\{k'_{0-7}\}_{k_{4-7}}$, 4). m_4 also encrypts and sends the keys to m_5 as $\{k'_{4-5}, k'_{4-7}, k'_{0-7}\}_{pk_5}$. As a result, all the members will get the new keys.

When a new member joins, only the keys along its path to the root have to be changed and distributed, which can be achieved in two rounds with atmost log_2n keys being changed.

F. Leave protocol

```
Step 1: Every member
      . updates the key tree by removing the leaving member node
      . updates the sponsor field appropriately along the leaving
      members path if required
      . determines the sponsor for changing keys along the leaving
      members path
Step 2: If sponsor of the leaving member is itself
      . generates new secret keys along the path and distributes
      them to co-distributors and directly to other members
Step 3: If co-distributor is itself
      . broadcasts the key sent by the leaving members sponsor by
      encrypting it with the appropriate key
```

Fig. 8. Leave Protocol.

Assume that member m_l leaves the group. Every member updates the key tree by deleting node m_l and updates the sponsor field along the path if required. Then they determine the sponsor who generates new keys along the leaving members path and distributes them. If the leaving member does not have a sibling then the first sponsor along the leaving members path becomes responsible for changing the keys along the leaving member's path (See Figure 8).

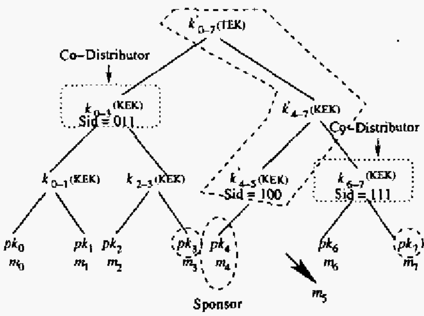

Fig. 9. A member m_5 leaves.

As shown in Figure 9, when a member m_5 leaves, all the members will remove the node and determine that m_4 is the sponsor of m_5. So m_4 initiates the rekeying process as follows: 1) generates new keys k'_{4-5}, k'_{4-7}, and k'_{0-7}. 2) after determining the co-distributors m_3 and m_7, encrypts as follows and broadcasts: $\{k'_{4-7}, k'_{0-7}\}_{pk_7}$, and $\{k'_{0-7}\}_{pk_3}$, 3). m_3 will decrypt k'_{0-7} and encrypt it as $\{k'_{0-7}\}_{k_{0-3}}$ and m_7 will decrypt k'_{4-7} and k'_{0-7} and encrypt them as $\{k'_{4-7}\}_{k_{6-7}}$ and $\{k'_{0-7}\}_{k_{4-7}}$, 4). As a result, all the members will get the new keys.

When a member leaves only the keys along its path to the root have to be changed and distributed, which can be achieved in two rounds with at most $log_2 n$ keys being changed.

G. Multiple join protocol

Suppose m new members join, they make a join request by broadcasting their public keys. The rightmost member in the key tree authenticates the new members, decides the locations for all the new members such that minimal number of keys are changed and broadcasts this information to other existing group members. The rightmost member also sends the virtual key tree and existing members public keys to the joining members. Every member upon receiving this message updates its key tree by adding m new nodes in the determined positions. In order to perform multiple joins in one aggregate operation, it is required to find the common keys shared by the joining members in an efficient way. To achieve that we use an already proposed scheme, an efficient and scalable key tree based dynamic conferencing scheme called KTDC in [52] which uses an efficient algorithm for computing the shared keys. There will be multiple sponsors responsible for changing the necessary keys. But here the shared keys which both sponsors have in common and which need to be changed will be changed by the rightmost sponsor among the sponsors (See Figure 10).

```
Step 1: Every member
      . updates key tree by adding new member nodes
      . updates the sponsor field along all the paths of the
      joining members
      . computes the keys that need to be changed
      . determines the sponsors who are responsible for changing
      these keys
Step 2: If sponsor for one of the joining members is itself
      . changes the secret keys along the joining members path
      and distributes them to co-distributors and directly
      to other members
      . if same key has to be changed, check if right sponsor is itself
      . if rightmost sponsor, change the key and distribute
Step 3: If co-distributor is itself
      . broadcasts the key sent by the joining members sponsor by
      encrypting it with the appropriate key
```

Fig. 10. Multiple Join Protocol.

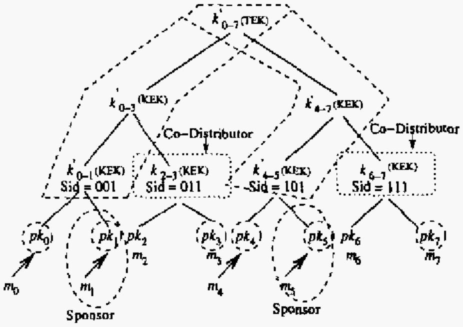

Fig. 11. New members m_0, m_1, m_4 and m_5 join.

As shown in Figure 11, when new members join, m_7 will determine the available positions (i.e., m_0, m_1, m_4, m_5) and place the members there. m_7 broadcasts this information to other group members. All members also know that m_5 is the sponsor of m_4 and m_1 is the sponsor of m_0. They also know that m_3 and m_7 are responsible for sending the key tree structure and the public key list to the joining members. m_5 initiates the rekeying process as follows: 1) generates new keys k'_{4-5}, k'_{4-7}, and k'_{0-7}. 2) after determining the co-distributors m_3 and m_7, encrypts as follows and broadcasts: $\{k'_{4-7}, k'_{0-7}\}_{pk_7}$, and $\{k'_{0-7}\}_{pk_3}$, 3). m_3 will decrypt k'_{0-7} and encrypt it as $\{k'_{0-7}\}_{k'_{0-3}}$ and m_7 will decrypt k'_{4-7} and k'_{0-7} and encrypt them as $\{k'_{4-7}\}_{k_{6-7}}$ and $\{k'_{0-7}\}_{k_{4-7}}$, 4). m_5 also encrypts and sends the keys to m_4 as $\{k'_{4-5}, k'_{4-7}, k'_{0-7}\}_{pk_4}$. Similarly m_1 regenerates the

0-7803-9290-6/05/$20.00 ©2005 IEEE.

keys along its path except for the root key which should be changed by the rightmost sponsor m_5. Both m_1 and m_5 do these operations in parallel. As a result, all the members whose keys have been changed will get the new keys.

Since all the operations are done in parallel, rekeying can be achieved in two rounds by all the sponsors.

When a network event causes all the previously occurred partitions to reconnect this is called a merge. Merge is similar to multiple join and this can also be achieved in two rounds which is better than that in TGDH.

H. Multiple leave protocol

```
Step 1: Every member
        . updates the key tree by removing all leaving member nodes
        . updates the sponsor field along all the leaving members
        paths if required
        . determines the sponsors responsible for changing the keys
        along the paths
Step 2: If sponsor for one of the leaving member is itself
        . generates new secret keys and distributes them to
        co-distributors and other members directly
        . if same key has to be changed, check if right sponsor is itself
        . if rightmost sponsor, change the key and distribute
Step 3: If co-distributor is itself
        . broadcasts the key sent by the leaving members sponsor by
        encrypting it with the common key
```

Fig. 12. Multiple Leave Protocol.

When multiple members leave, every member updates its key tree by deleting those member nodes and the sponsor fields along all the paths. Then they determine the keys that need to be changed and the sponsors responsible for changing those keys. There will be multiple sponsors and each sponsor regenerates the keys and distributes them. If two sponsors are responsible for changing the same key then the rightmost among the sponsors will change the key (See Figure 12).

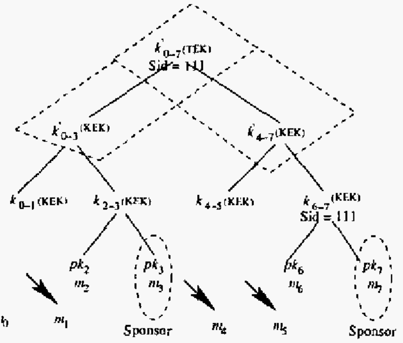

Fig. 13. Members m_0, m_1, m_4 and m_5 leave

As shown in Figure 13, when several members m_0, m_1, m_4 and m_5 leave, every member updates its key tree by deleting those member nodes. Every member also determines that m_3 and m_7 are the sponsors. m_7 initiates the rekeying process as follows: 1) generates new keys k'_{4-7}, and k'_{0-7}. 2) encrypts the new keys as follows and broadcasts: $\{k'_{4-7}, k'_{0-7}\}_{k_{6-7}}$, and $\{k'_{0-7}\}_{pk_3}$, 3) m_3 will decrypt k'_{0-7} and encrypt it as $\{k'_{0-7}\}_{k'_{0-3}}$ and broadcasts it. Similarly m_3 generates the keys k'_{0-3} and encrypts it with k_{2-3} and broadcasts it. Both m_3 and m_7 do these operations in parallel. As a result, all the members whose keys have been changed will get the new keys.

In case of a network failure which causes disconnectivity, the group gets split and this partition can be dealt with as a multiple leave operation. Thus, even for network partition the protocol requires only two rounds for regenerating and distributing the keys. This is a great improvement compared to TGDH which requires several rounds.

I. Authentication in DGKD

Most CGKA protocols do not contain an authentication component. Furthermore, the authenticated CGKA protocols [53], [54], [55], [56], [57] are non-scalable and/or non-dynamic. In contrast, the new DGKD protocol is not only scalable and dynamic but also able to provide easy and strong authentication. Consider two scenarios: (1) the sponsor m_4 transmits a new key k'_{0-7} to a co-distributor m_3. (2) m_3 transmits the key k'_{0-7} to members m_0,m_1,m_2 who are in the responsibility scope of m_3. In the first case, m_4 signs the key k'_{0-7}(using m_4's private key), encrypts both k'_{0-7} and the signed k'_{0-7}(using m_3's public key pk_3), and sends the result to m_3. m_3 after receiving the message, decrypts k'_{0-7} and then verifies m_4's signature. In the second case, m_3 signs k'_{0-7}(by its private key), encrypts both k'_{0-7} and the signed k'_{0-7}(using k_{0-3} which covers m_0 to m_3). Then each of the members from m_0 to m_2 can verify m_3's signature.

IV. Discussions

We discuss the performance and security of our protocol in this section and analyze the communication and computation costs for join, leave, multiple join and multiple leave operations. Tree based Group Diffie-Hellman (TGDH) [19], [34] is one of the most typical CGKA protocols in terms of efficiency and scalability, so we focus on the comparison between DGKD and TGDH.

Key generation is independent, i.e., only the sponsor is involved, thus there is no need for synchronization with other members which is required in TGDH. In this sense, DGKD is more resilient to network congestion, delay and failure than TGDH. DGKD also has strong yet simple authentication. It is also collusion free because the new keys are independent of the old keys and no matter how many members collude they cannot get the keys. Thus, it is unconditionally secure. Both TGDH and DGKD require

0-7803-9290-6/05/$20.00 ©2005 IEEE.

two rounds for single join and leave operations. As for multiple join and leaving operations, DGKD requires two rounds but TGDH requires $log(p)$ rounds where p is the number of members involved. DGKD uses public key encryption for sending the keys to co-distributors and secret key encryption for further distribution of keys (from the co-distributors to the members). TGDH requires performing modular exponentiations which is in the same complexity as the public key encryption. In summary, DGKD is comparable and in some cases better than TGDH in terms of communication and computation costs.

V. Conclusion

We proposed a new class of GKM protocols for SGC with strong yet simple authentication capability. The proposed protocol solves some serious problems in the existing protocols and is simple, robust, efficient, and scalable. The future work is to implement and test the new protocol.

References

[1] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," *Proceedings of INFOCOM'99: Conference on Computer Communications*, vol. 2, pp. 708–716, Mar. 1999.

[2] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," *Lecture Notes in Computer Science (Advances in Cryptology-EUROCRYPT'99)*, vol. 1592, pp. 459–470, 1999.

[3] G. Caronni, K. Waldvogel, D. Sun, and B. Plattner, "Efficient security for large and dynamic multicast groups," *Proceedings of the Seventh IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE '98) (Cat. No.98TB100253)*, pp. 376–383, June 1998.

[4] H. Hamey and C. Muckenhim, "Group Key Management Protocol (GKMP) Architecture," *RCF 2094*, July 1997.

[5] H. Hamey and C. Muckenhim, "Group Key Management Protocol (GKMP) Specification," *RCF 2093*, July 1997.

[6] H. Harney and E. Harder, "Logical key hierarchy protocol," *Internet Draft (work in progress), draft-harney-sparta-lkhp-sec-00.txt, Internet Engineering Task Force*, Mar. 1999.

[7] G. Noubir, "Multicast security," *European Space Agency, Project: Performance Optimization of Internet Protocol Via Satellite*, Apr. 1998.

[8] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," *Internet Draft (work in progress), draft-wallner-key-arch-01.txt, Internet Eng. Task Force*, Sept. 1998.

[9] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *SIGCOMM '98, Also University of Texas at Austin, Computer Science Technical report TR 97-23*, pp. 68–79, Dec. 1998.

[10] L. R. Dondeti, S. Mukherjee, and A. Samal, "A dual encryption protocol for scalable secure multicasting," *In Fourth IEEE Symposium on Computers and Communications*, pp. 2–8, July 1999.

[11] F. Du, L. M. Ni, and A. H. Esfahanian, "Towards solving multicast key management problem," *ICCCN'99 Eighth International Conference on Computer Communications and Networks*, pp. 232–236, Oct. 1999.

[12] S. Mittra, "Iolus: A framework for scalable secure multicasting," *Journal of Computer Communication Reviews*, vol. 27, no. 4, pp. 277–288, 1997.

[13] R. Molva and A. Pannetrat, "Scalable multicast security in dynamic groups," *6th ACM Conference on Computer and Communications Security (ACM CCS 1999), Singapore*, pp. 101–112, Nov. 1999.

[14] M. Burmester and Y. Desmedt, "Efficient and secure conference-key distribution," *Security Protocols Workshop*, pp. 119–129, 1996.

[15] L. R. Dondeti, "Efficient private group communication over public networks," *Phd. Dissertation, CSE UNL*, 1999.

[16] I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," *IEEE Transactions on Information Theory*, vol. 28, pp. 714–720, Sept. 1982.

[17] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A secure audio teleconference system," *Advances in Cryptology-CRYPTO'88, LNCS, Springer-Verlag*, vol. 403, pp. 520–528, Aug. 1990.

[18] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," *ACM Conference on Computer and Communications Security (ACM CCS 1996)*, pp. 31–37, Mar. 1996.

[19] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," *In Proceedings of the 7th ACM Conference on Computer and Communications Security (ACM CCS 2000)*, pp. 235–244, Nov. 2000.

[20] Y. Kim, A. Perrig, and G. Tsudik, "Communication-efficient group key agreement," *In Information System Security, Proceedings of the 17th International Information Security Conference IFIP SEC'01*, pp. 229–244, June 2001.

[21] A. Bakkardie, "Scalable multicast key distribution," *RFC 1949*, 1996.

[22] A. Beimel and B. Chor, "Interaction in key distribution schemes," *Advances in Cryptology - CRYPTO'93, LNCS, Springer, Berlin*, vol. 773, pp. 444–457, 1994.

[23] A. Beimel and B. Chor, "Communications in key distribution schemes," *IEEE Transactions on Information Theory*, vol. 42, pp. 19–28, 1996.

[24] R. Blom, "An optimal class of symmetric key generation systems," *Advances in Cryptology - EUROCRYPT'84, LNCS, Springer, Berlin*, vol. 209, pp. 335–338, 1985.

[25] C. Blundo and A. Cresti, "Space requirements for broadcast encryption," *Advances in Cryptology - EUROCRYPT'94, LNCS, Springer, Berlin*, vol. 950, pp. 287–298, 1995.

[26] C. Blundo, L. A. F. Mattos, and D. R. Stinson, "Generalized Beimel-Chor scheme for broadcast encryption and interactive key distribution," *Theoretical Computer Science*, vol. 200, pp. 313–334, June 1998.

[27] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfect secure key distribution for dynamic conferences," *Advances in Cryptology - CRYPTO'92, LNCS, Springer, Berlin*, vol. 740, pp. 471–486, Aug. 1993.

[28] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," *Advances in Cryptology - EUROCRYPT'94, LNCS, Springer, Berlin*, vol. 950, pp. 275–286, May 1995.

[29] I. F. Bob Briscoe, "Nark: receiver-based multicast non-repudiation and key management," *Proceedings of the 1st ACM conference on Electronic commerce*, pp. 22–30, Nov. 1999.

[30] Y. Challal, H. Bettahar, and A. Bouabdallah, "Sakm: a scalable and adaptive key management approach for multicast communications," *ACM SIGCOMM Computer Communication Review*, vol. 34, pp. 55–70, Apr. 2004.

[31] W. Chen and L. R. Dondeti, "Recommendations in using group key management algorithms," *DARPA Information Survivability Conference and Exposition*, vol. 2, pp. 222–227, Apr. 2003.

[32] A. Fiat and M. Naor, "Broadcast encryption," *Advances in Cryptology - CRYPTO'93, LNCS, Springer, Berlin*, vol. 773, pp. 480–491, 1994.

[33] S. M. Ghanem and H. Abdel-Wahab, "A secure group key management framework: Design and rekey issues," *Eighth IEEE International Symposium on Computers and Communications*, pp. 797–802, June 2003.

[34] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information Systems Security*, vol. 7, pp. 60–96, Feb. 2004.

[35] F.-Y. Lee and S. Shieh, "Scalable and lightweight key distribution for secure group communications," *International Journal of Network Management*, vol. 14, pp. 167–176, May 2004.

[36] J.-C. Lin, C.-Y. Chou, F. Lai, and K.-P. Wu, "A distributed key management protocol for dynamic groups," *27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, pp. 0113–0122, Nov. 2002.

[37] D. Liu, P. Ning, and K. Sun, "Cryptographic protocols/ network security: Efficient self-healing group key distribution with revocation capability," *Proceedings of the 10th ACM conference on Computer and communication security*, pp. 231–240, Oct. 2003.

[38] C. Meadows and P. Syverson, "Group key management and signatures: Formalizing gdoi group key management requirements in npatrl," *Proceedings of the 8th ACM conference on Computer and Communications Security*, pp. 235–244, Nov. 2001.

[39] C.-S. Park and D.-H. Lee, "Secure and efficient key management for dynamic multicast groups," *ACM SIGOPS Operating Systems Review*, vol. 35, pp. 32–38, Oct. 2001.

[40] S. Rafaeli and D. Hutchison, "Hydra: A decentralised group key management," *Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02)*, pp. 62–67, June 2002.

[41] A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," *IEEE transactions on Software Engineering*, vol. 29, pp. 444–458, May 2003.

[42] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, pp. 769–780, Aug. 2000.

[43] D. R. Stinson, "On some methods for unconditionally secure key distribution and broadcast encryption," *Design, Codes and Cryptography*, vol. 12, pp. 215–243, June 1997.

[44] Y.-M. Tseng, "A scalable key-management scheme with minimizing key storage for secure group communications," *International Journal of Network Management*, vol. 13, pp. 419–425, Nov. 2003.

[45] S. Zhu, S. Setia, and S. Jajodia, "Performance optimizations for group key management schemes," *23rd International Conference on Distributed Computing Systems*, pp. 163–171, May 2003.

[46] G. H. Chiou and W.T.Chen, "Secure broadcasting using the Secure Lock," *IEEE Transactions on Software Engineering*, vol. 15, pp. 929–934, Aug. 1989.

[47] D. R. Stinson, ed., *Cryptography: Theory and Practice*. Boca Raton, Florida, USA: CRC Press, Inc., 1995.

[48] X. Zou, B. Ramamurthy, and S. S. Magliveras, eds., *Secure Group Communications over Data Networks*. New York, NY, USA, ISBN: 0-387-22970-1 (The ebook ISBN: 0-387-22971-X): Springer, Oct. 2004.

[49] L. R. Dondeti, S. Mukherjee, and A. Samal, "DISEC: a distributed framework for scalable secure many-to-many communication," *In Proceedings of Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, pp. 693–698, July 2000.

[50] X. B. Zhang, S. S. Lam, D.-Y. Lee, and Y. R. Yang, "Protocol design for scalable and reliable group rekeying," *Proceedings SPIE Conference on Scalability and Traffic Control in IP Networks*, pp. 87–108, Aug. 2001.

[51] X. Zou and B. Ramamurthy, "A block-free tree-based group Diffie-Hellman key agreement for secure group communications," *Proceedings of International Conference on Parallel and Distributed Computing and Networks, Innsbruck, Austria*, pp. 288–293, Feb. 2004.

[52] X. Zou, S. Magliveras, and B. Ramamurthy, "Key tree based scalable secure dynamic conferencing schemes," *Proceedings of International Conference on Parallel and Distributed Computing and Systems (PDCS 2004), MIT Cambridge, MA, USA, November 9-11*, pp. 61–66, Nov. 2004.

[53] G. Ateniese, M. Steiner, and G. Tsudik, "Authenticated group key agreement and friends," *Proceedings of the 5th ACM conference on Computer and communications security, CA, USA*, pp. 17–26, 2008.

[54] E. Bresson, O. Chevassut, and D. Pointcheval, "Dynamic group Diffie-Hellman key exchange under standard assumptions," *Advances in Cryptology - EUROCRYPT'2002, LNCS, Springer, Berlin*, vol. 2332, pp. 321–336, 2002.

[55] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater, "Provably authenticated group Diffie-Hellman key exchange," *Proc. 8th Annual ACM Conference on Computer and Communications Security (CCS'01)*, pp. 255–264, 2001.

[56] J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," *The 23rd Annual International Cryptology Conference, CRYPTO 2003, CA, USA*, Aug. 2003.

[57] H.-J. Kim, S.-M. Lee, and D. H. Lee, "Constant-round authenticated group key exchange for dynamic groups," *Asiacrypto'04, http://www.iris.re.kr/ac04/data/Asiacrypt2004 /06%20Key%20Management/02_Hyun-Jeong%20Kim.pdf*, 2004.