

# Mykil: A Highly Scalable Key Distribution Protocol for Large Group Multicast

Jyh-How Huang and Shivakant Mishra  
Department of Computer Science  
University of Colorado, Campus Box 0430  
Boulder, CO 80309-0430, USA.  
Email: {huangjh|mishras}@cs.colorado.edu

**Abstract**—This paper describes the design, implementation, and evaluation of Mykil, which is a new key distribution protocol for secure group multicast. Mykil has been designed to be scalable to large group sizes. It is based on a combination of group-based hierarchy and key-based hierarchy systems for group key management. Important advantages of Mykil include a fast rekeying operation for large group sizes, continuous availability of the key management service in a disconnected network environment, an ability to map the group structure to the underlying network infrastructure, robustness, support for user mobility, and support for smaller hand-held devices.

## I. INTRODUCTION

A large number of Internet applications need group multicast support. Examples include pay-per-view programs, video-on-demand services, stock quote updates, and video conferencing. These applications rely on the existence of a secure multicast service built on top of IP multicast. A secure multicast service implements a *secure multicast group* in which members register and authenticate themselves with a designated registration authority, receive a set of *cryptographic key(s)*, and use these keys to encrypt the multicast data that they send and decrypt the multicast data that they receive. A key management server manages a set of cryptographic keys used for various purposes in a secure multicast group. It stores these keys, updates them when certain events occur, and distributes them to the group members using a key distribution protocol. The process of updating the cryptographic keys, and distributing them to the group members is called a *rekeying* operation. Rekeying is required in secure multicast to ensure that only the *current* group members can send encrypted multicast data, and decrypt the received multicast data.

In this paper, we propose a new protocol called Mykil (Multi-Hierarchy Based Key Distribution Protocol) for managing cryptographic keys in large multicast groups (100,000 members or more) exhibiting frequent membership changes. The cost of key management in large groups can become prohibitively expensive. Mykil cleverly combines two different types of hierarchy schemes—group-based hierarchy and key-based hierarchy, to provide scalable solution for key management in large multicast groups.

Mykil makes four important contributions. First, it provides a very fast rekeying operation by ensuring that key updates take place at only a small number of group members during a member join or leave event. The cost of rekeying operation

is further reduced by batching member join and leave events. Second, it is designed to support group members that access a multicast service using small devices such as PDAs or cell phones that have limited resources. This is done by minimizing the memory, bandwidth, and CPU requirements for key management functions at different group members. Third, Mykil is designed to support both static and mobile group members. Finally, the design of Mykil ensures that the key management functionality is robust, and remains available to all group members even when the underlying communication network partitions.

## II. RELATED WORK

Current key management protocols can be divided into two categories [5]: (1) group-based hierarchy schemes, and (2) key-based hierarchy schemes. Group-based hierarchy schemes [3], [10], [6], [7] address the scalability issue by organizing a multicast group into a hierarchy of subgroups. The basic idea is to distribute the functionality of the key management service among the subgroups, and thereby achieve decentralization and scalability. Key-based hierarchy schemes [12], [11], [9], [1], [4] on the other hand address the problem of scalability by organizing a tree-structured hierarchy of cryptographic keys. Scalability is achieved in these protocols by reducing the number of messages exchanged during a rekeying operation.

Both group-based and key-based hierarchy protocols have their advantages and disadvantages. Both types of protocols provide high scalability over a naive key distribution protocol. Because of decentralization of key management functionality, a protocol based on group-based hierarchy can tolerate network partitions. A protocol based on key-based hierarchy on the other hand cannot tolerate network partitions. Another advantage of group-based hierarchy is that the actual organization of different areas can be mapped quite well to the underlying network infrastructure. For example, all members located within a subnet or an organization may belong to one area. Storage requirement for users in a group-based hierarchy is extremely low (two or three keys), while it is  $O(\log n)$  in key-based hierarchy. Finally, key distribution during a rekeying operation in a group-based hierarchy protocol relies on subgroup controllers to send separate messages to all subgroup members. This can result in a subgroup controller becoming a performance and scalability bottleneck.

### III. MYKIL: PROTOCOL DETAILS

The main motivation of Mykil is to combine the two hierarchy schemes in such a way that the good features of the two schemes are preserved and the limitations of the two schemes are eliminated. Although such a combination has been hinted in [6] and [5], no protocol based on this combination has been developed so far. Mykil is the 1st protocol that exploits this idea and provides support for scalability, performance, robustness, and support for mobility and small devices.

Mykil is based on Iolus[6] and LKH[12]. It uses the idea of group-based hierarchy of Iolus to divide a multicast group into several smaller subgroups called *areas* with a designated area controller (AC) for each area. There is a separate *area key* for each area. Different areas are linked with one another to form a tree structure, with ACs providing the links—an AC of an area  $A$  is also a member of another area  $B$  (area  $B$  is  $A$ 's parent in the tree-structure organization). A group member belongs to exactly one area. Like LKH, Mykil builds a tree-structured hierarchy of cryptographic keys called *auxiliary-key tree* in each area to facilitate key distribution to the area members. The area controller of an area serves as the root of the auxiliary-key tree of that areas, and each member of this area is associated with a different leaf of this auxiliary-key tree. Multicast data propagation in Mykil is identical to the multicast data propagation in Iolus. A group member multicasts data by encrypting it using a random key. This random key is encrypted using the member's area key and appended to the multicast message. To forward multicast data (that has been encrypted using a random key) to another area, an AC decrypts the random key, and reencrypts it using the other area's area key. An example of data propagation initiated by member  $m_s$  is shown in Figure 1.

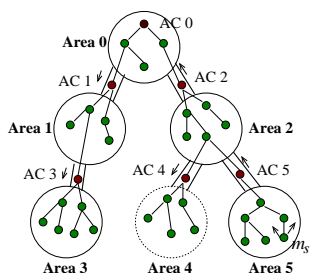


Fig. 1. Organization of group members and data propagation in Mykil.

#### A. Group and Area Creation

A multicast group  $\mathcal{G}$  is initialized by creating a root area with a designated group controller for this area. This root area forms the root in the tree-structured organization of the all areas comprising  $\mathcal{G}$ . The area controller of this root area is also the group controller of  $\mathcal{G}$ . In terms of functionalities, a group controller is no different from any other area controller of the group. Responsibilities of an area controller include: (1) managing cryptographic keys of its area, (2) forwarding

multicast data as shown in Figure 1, (3) managing authorization information to determine who can join the group, (4) maintaining the auxiliary key tree of its area, and (5) managing member join and leave events.

Creation of a new area in  $\mathcal{G}$  is initiated by a designated area controller  $A_c$ . Before creating a new area,  $A_c$  must obtain an authorization information database  $\mathcal{AI}$  needed to determine who can become a member of  $\mathcal{A}$  (or  $\mathcal{G}$  in general).  $A_c$  chooses another area to be the parent of its area. This choice can be based on network proximity, administrative policy, or other criteria.  $A_c$  then joins the chosen parent area (as a regular group member) by contacting the area controller of the parent area as described in the next subsection.

#### B. Member Join

An entity  $m$  that wishes to join  $\mathcal{G}$  chooses an appropriate area controller  $A_c$ , and unicasts a *join request* containing all the necessary authorization information to  $A_c$  through a secure channel. On receiving this join request,  $A_c$  checks the authorization information. The rest of the join protocol is similar to a join protocol in LKH.  $A_c$  creates a new area key  $K'_a$  and multicasts  $E_{K_a}(K'_a)$  (encryption of  $K'_a$  using the previous area key  $K_a$ ) to all current area members. In addition,  $A_c$  determines an appropriate (empty) leaf position in the auxiliary key tree of its area to place  $m$ . Area controllers in Mykil maintain a balanced tree structure of auxiliary keys such that each node (except leaf) in this tree has up to four children. This is based on the observation that a tree structure with each node having four children provides the best overall performance [12]. If an empty leaf position is found in the current auxiliary-key tree,  $m$  is placed at that position, and  $A_c$  unicasts a message to  $m$  containing the area key and all the auxiliary keys along the path from  $m$  to the root of the auxiliary-key tree. However, if no empty leaf position is present in the current auxiliary-key tree, a new level is created at the shallowest, left-most leaf node as shown in Figure 2.

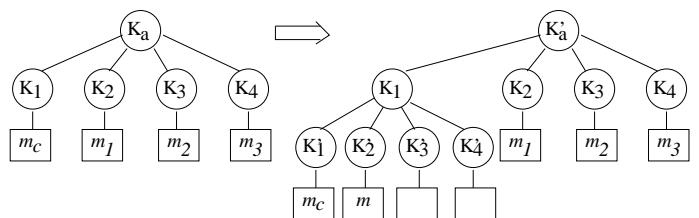


Fig. 2. A new member  $m$  joins an area.

#### C. Member Leave

The members leave protocol is again similar to that in LKH, except for one important step. Since the join operation is much less expensive if an empty leaf is already present in the tree, Mykil increases the likelihood of this scenario by not pruning the leaf after a member leaves. All keys along the path from the node corresponding to the leaving member to the root are changed, and the changed keys are multicast by

encrypting them using appropriate auxiliary keys. An example when member  $m_1$  decides to leave is shown in Figure 3 (shown as a binary tree here for simplicity). Mykil keeps the size of membership of an area relatively small, e.g. maximum 5000 members. This requires area members to store atmost 11 auxiliary keys, which is quite reasonable for smaller devices.

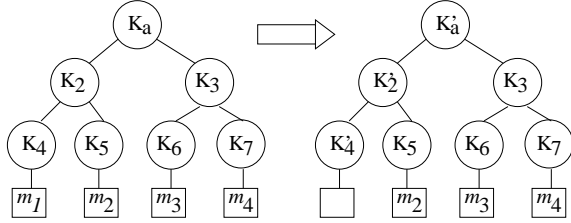


Fig. 3. Member  $m_1$  requests to leave the group.

#### D. Batching

The main idea of batching [1], [13], [10] is to perform a rekeying operation only after a minimum number of member join/leave requests have been received, and/or a certain time interval has elapsed. Mykil employs batching to reduce the overhead of rekeying operations in three ways: (1) aggregation of join events, (2) aggregation of leave events, and (3) aggregation of join and leave events.

When a new join request from a new member  $m$  arrives at  $A_c$ ,  $A_c$  simply calculates a new leaf position for  $m$ , records the identity of  $m$  and other current members whose path may have changed due to this join, and sets an *update needed* flag. When a new multicast data packet arrives at  $A_c$ ,  $A_c$  checks the update needed flag. If set, it multicasts a new area key to the current area members, sends appropriate unicast messages to the members whose identities were recorded, resets the update needed flag, and then forwards the multicast data. In similar manner, all consecutive leave events are aggregated until a new multicast data packet arrives at  $A_c$ . Finally, both join and leave events are aggregated in Mykil. The procedure to do so is essentially a union of the join aggregation and leave aggregation procedures with minor changes.

To reduce the complexity of rekeying operation, and preserve freshness of the area key, Mykil performs a rekeying operation under two conditions: (1) when a new multicast data packet is received by an area controller and the update needed flag is set, or (2) when some time interval has elapsed since the last rekeying operation.

#### E. Member Mobility and Fault Tolerance

An important aspect of Mykil is that it is designed to be robust, support mobile group members, and support operation in a disconnected environment. Common failures such as communication partitions and node crashes can result in a loss of communication between a group member and the area controller of its area. Mobility of a group member can result in either a loss of communication, or degraded communication between the member and its area controller. The decentralized

nature of Mykil allows operation in a disconnected environment. As long as a member can contact its area controller, it can continue to multicast data and receive data multicast by another member with in the same partition of the network. However, if a member loses contact with its area controller, it can neither receive, nor send any multicast data. In a nutshell, fault-tolerance support in Mykil consists of three parts:

- 1) When a group member detects that it can no longer communicate with its area controller, it attempts to join another area by contacting that area's area controller.
- 2) When an area controller detects that it can no longer communicate with one of its area member, it terminates the membership of that member from its area.
- 3) Finally, when an area controller detects that it can no longer communicate with the area controller of its parent area, it attempts to change its parent area by contacting that area's area controller.

To enable group members and area controllers detect communication problems, area controllers of each area multicast *alive* messages with in their respective areas whenever they encounter an idle period. An idle period occurs at an area controller when it hasn't multicast any message in its area in the last  $T_{idle}$  time units. In addition, each group member sends an *alive* message to its area controller whenever it determines that it has not sent any message to its area controller in the last  $T_{active}$  time units. Typically the value of  $T_{active}$  is much larger than the value of  $T_{idle}$ . Based on this, each member or area controller can implement its own criteria to decide if it has been disconnected. For example, a member can decide that it has been disconnected from its area controller, if it has not received any message from it in the last  $5 * T_{idle}$  time units. Similarly, an area controller can decide that one of its area member has been disconnected, if it does not receive any message from that member in the last  $5 * T_{active}$  time units.

When a member determines that it has been disconnected from its area controller, it attempts to join another area. To do so, the member sends its authentication credentials to another area controller that it can communicate with. The join procedure is similar to the one described in Section III-B. However, the member authentication procedure is slightly different. Since the joining member in this case was a member of another area, it should not be required to present all credentials that it needed to when joining the group initially. For example, if the supported application charges the users to become group members, a member that is only changing its area should not be charged again. To implement this, Mykil employs a mechanism similar to Kerberos [8]. An area controller provides a user a *membership ticket* when the user joins the group. This ticket is valid for a particular time period, and encrypted using a secret key that is known to only area controllers. When a user attempts to join another area, it presents its membership ticket to the area controller. An area controller checks this membership ticket before granting membership to the rejoining user.

When an area controller determines that it can no longer communicate with one of its area member, it terminates the membership of that member from its area. It essentially executes the protocol described in Section III-C. While it is not strictly required to terminate the membership of such a member, the membership is terminated to keep the size of auxiliary-key tree small.

Finally, when an area controller  $A_c$  detects that it can no longer communicate with the area controller of its parent area, it attempts to change its parent area by contacting another area's area controller. To do so, each area controller maintains a list of one or more preferred area controllers.  $A_c$  chooses an appropriate area controller from this list, and sends a join request to that area controller. The rest of the protocol is same as described in Section III-B.

#### IV. ANALYSIS

We analyze the resource requirements for the users in Mykil and compare them with Iolus and LKH. We consider three types of resources: (1) storage, (2) CPU, and (3) bandwidth consumption for join and leave events. We also analyze the gains of aggregating join and leave events in Mykil.

##### A. Storage Requirements

Since Mykil is designed to support smaller, hand-held devices such as cell phones and palmtops that have limited storage capabilities, it is important that storage requirements be minimized for the users. For a group with 100,000 members, a member in LKH will need to store about 17 keys, where 17 is the height of the balanced binary tree with 100,000 leaves. On the other hand, a member in Iolus will only need to store 2 keys, an area key and a pairwise secret key with area controller. Assuming that we limit the membership size of an area to about 5000 members in Mykil, a member in Mykil will need to store about 11 keys. The size of a key should be large enough to ensure the group key secrecy requirements. We use fastest cipher algorithm in OpenSSL, RC4, with 128 bits keys in our implementation. This means that a user needs 32 bytes in Iolus, 272 bytes in LKH, 176 bytes in Mykil to store the required keys.

##### B. CPU Requirements

For a join event, the computational requirements at the joining member are similar in all three protocols. The joining member receives the new area/group key and some auxiliary keys. However, a join event requires existing members to do some computation as well. In particular, group key of all members is updated in LKH, while area key of the members of only one area is updated in Iolus and Mykil. So, on an average, the CPU requirements are larger in LKH compared to Iolus or Mykil during a join event.

For a leave event, each member of one area will receive a new area key in Iolus. For a group of 100,000 members with maximum area size of 5000 members, 5000 members will update one key. In case of LKH, on an average, 50% of members will need to update one key, 25% will update two

keys, 12.5% will update three keys, 6.25% will update four keys, and so on. For a group of 100,000 members, this implies that 50,000 members will update one key, 25,000 members will update two keys, 12,500 members will update three keys, 6,250 members will update four keys, and so on. Finally, in Mykil, only the members with in one area are affected. For an area of 5000 members, 2500 members will update one key, 1250 members will update two keys, 625 members will update three keys, 313 members will update four keys, and so on.

##### C. Bandwidth Consumption

The bandwidth consumption per group member during a rekeying operation depends on the length of the key update message. For a join event, the length of key update message that is multicast is same in all three protocols, i.e. the length of the encrypted new group/area key. In addition, LKH and Mykil also unicast the key path to the new member. In Mykil, this corresponds to  $16 \cdot 12 = 172$  bytes in an area of 5000 members. In LKH, this corresponds to  $16 \cdot 17 = 272$  bytes for a group of 100,000 members.

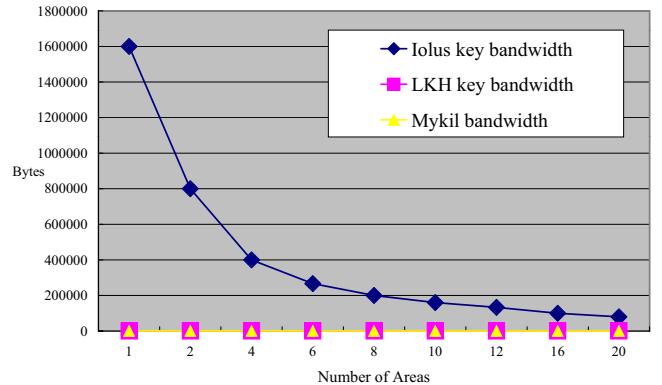


Fig. 4. Bandwidth consumption during a leave event.

For a leave event, the length of key update message in Iolus depends on the area size. For an area of 5000 members and assuming 128-bit keys, the length of this message will be about 80,000 bytes. In LKH and Mykil, the size of rekeying message during a leave event depends on the height of the tree. In particular, an updated key corresponding to a node  $n$  is encrypted separately by keys corresponding all children of  $n$ . Since, all keys along the path from the root to the leaf corresponding to the leaving member are updated, this implies a rekeying message of  $2 \cdot 17 \cdot 16 = 544$  bytes in LKH (100,000 members in the group), and  $2 \cdot 12 \cdot 16 = 384$  bytes in Mykil. Figure 4 shows the bandwidth requirements for the three protocols, and Figure 5 shows in detail the bandwidth requirements in Mykil and LKH. These graphs show clearly that the bandwidth requirements of Mykil and LKH is quite low compared to the bandwidth requirements of Iolus. Bandwidth requirement in Mykil is further reduced by aggregating consecutive join or leave events. For example, Figure 6 shows the reduction in Mykil by aggregating ten consecutive leave events.

## VI. CONCLUSION

A large number of emerging Internet applications featuring large group sizes are based on secure group multicast model. Management of cryptographic keys has turned out to be the main bottleneck in supporting these applications. Building hierarchy in key management system has been the basis for building scalable key management systems. In this paper, we have proposed a new protocol, Mykil, which combines two hierarchy schemes (LKH and Iolus) in such a way that all important advantages of the two schemes - scalability, mapping to the underlying network infrastructure, operation in a disconnected environment - are retained. Mykil improves on LKH by reducing the resource requirements for a group member, providing support for operation in a disconnected environment, and providing an ability to map the group organization to the underlying network infrastructure. Mykil improves on Iolus by reducing the bandwidth requirements and eliminating the performance bottleneck of area controller. Mykil also supports smaller, hand-held devices, user mobility, and robustness. An analysis shows that the resource requirements for a group member in Mykil are reasonable, and a client can avail of this protocol via smaller devices. Performance measurements of our prototype show that Mykil is fast, scalable, and has lower resource requirements at the user nodes.

## ACKNOWLEDGEMENT

Lianjun Jiang, Jing Deng and Wang Ting Lin helped in developing prototype of Mykil. We appreciate their contribution.

## REFERENCES

- [1] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Daha. Key management for secure Internet multicast using boolean function minimization technique. *ACM SIGCOMM'99*, March 1999.
- [2] W. Fenner. Internet group management protocol (IGMP). Request For Comments 2236, Xerox PARC, November 1997.
- [3] T. Hardjono, B. Cain, and I. Monga. Intra-Domain group key management protocol (IGKMP). Internet draft, IETF, February 2000.
- [4] D. McGrew and A. Sherman. Key establishment in large dynamic groups using one-way function trees, May 1998. Available at <http://www.cs.umbc.edu/~sherman/Papers/itse.ps>.
- [5] S. Mishra. Key management in large group multicast. Technical Report CU-CS-940-02, Department of Computer Science, University of Colorado, Boulder, CO., 2002.
- [6] S. Mitra. Iolus: A framework for scalable secure multicasting. In *Proceedings of the ACM SIGCOMM'97*, September 1997.
- [7] R. Molva and A. Pannetrat. Scalable multicast security in dynamic groups. In *Proceedings of the 6th ACM Conference on Computer and Communication Security*, November 1999.
- [8] C. Neuman and T. Theodore. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9), September 1994.
- [9] A. Perrig, D. Song, and J. Tygar. ELK, a new protocol for efficient large-group key distribution. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, 2001.
- [10] S. Setia, S. Koussih, and S. Jajodia. Kronos: A scalable group re-keying approach for secure multicast. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 2000.
- [11] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. Request For Comments (Informational) 2627, Internet Engineering Task Force, June 1999.
- [12] C. Wong, M. Gouda, and S. Lam. Secure group communication using key graphs. In *Proceedings of the ACM SIGCOMM'98*, October 1998.
- [13] R. Yang, S. Li, B. Zhang, and S. Lam. Reliable group rekeying: A performance analysis. *ACM SIGCOMM'01*, August 2001.

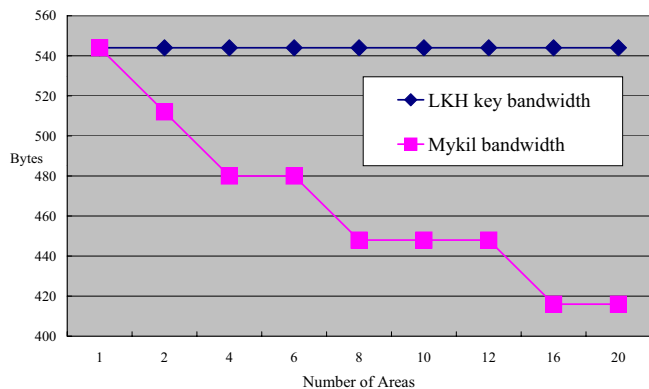


Fig. 5. Bandwidth consumption in Mykil and LKH during a leave event.

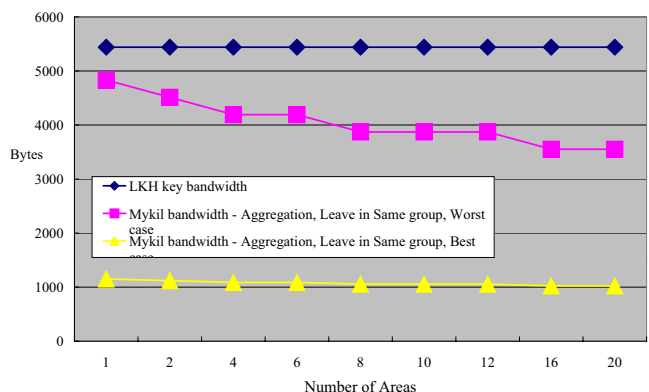


Fig. 6. Bandwidth consumption in Mykil with and without aggregation of leave events.

## V. IMPLEMENTATION AND PERFORMANCE

We implemented a prototype of Mykil that runs on Linux. The implementation uses IP multicast in LAN, and TCP to forward data/control information cross areas. We have experimented with this prototype over a LAN and WAN. To measure the performance, we conducted an experiment over a network of 5 PCs (Compaq PIII-800, 256 MB) running Suse 8.1 connected by a 10 Mbps Ethernet. We constructed a multicast group of size 50 with 5 areas— A, B, C, D, and E, each containing 10 members. A was the root area with B and C as its children, D was a child of B, and E was a child of C. We used keys of 64 bits in this experiment. A single join operation results in key updates in only 10 members in Mykil in this scenario, while all 50 members have to update their keys if LKH is used instead. To measure the effect of batching, we had ten new members join the area E (which was empty in the beginning). When batching is used, only one packet of size 24 B was multicast. On the other hand if batching is not used, a total of 55 packets, each 24 B long, need to be multicast.