
Key Management Approaches to Offer Data Confidentiality for Secure Multicast

Kin-Ching Chan and S.-H. Gary Chan,
The Hong Kong University of Science and Technology

Abstract

Multicasting is an efficient way to deliver data to a large group of users in applications such as Internet stock quotes, audio and music delivery, file and video distribution, etc. Many of these applications require the security feature of data confidentiality, which is not readily offered by the “open” nature of multicast. In order to offer such confidentiality, the encryption and decryption keys must be constantly changed upon a membership change. In this article, after discussing some performance criteria to offer secure multicast, we present a number of the proposed key management schemes for data confidentiality. We categorize these schemes into four groups: key tree-based approaches, contributory key agreement schemes supported by the Diffie-Hellman algorithm, computational number theoretic approaches, and secure multicast framework approaches. Through examples, we describe the operation of the schemes and compare their performances.

Multicasting is an efficient way to deliver data from a sender to multiple receivers [1]. One of the main advantages of multicast over unicast is the reduction of the network resources sent to the receivers. As opposed to broadcasting, where everyone receives the data, multicast delivers data only to a group of interested users termed a “multicast group.” At any time users may “leave” or “join” the group. Multicast protocols achieve low network transmission overheads and high user scalability by building an efficient multicast tree dynamically upon each user leaving or joining (by means of pruning or grafting). There are a number of exciting multimedia applications that make good use of multicast capability, such as stock quote services, video-conferencing, pay-per-view TV, Internet radio, and so on. Many of these multicast applications require security in data transmission, i.e., data can only be exchanged among an exclusive group of users.

Traditional security measures are mainly applicable to a unicast environment. For instance, data confidentiality, one of the most important features in network security, can be offered in this environment by means of a pair of keys. To offer the same feature in multicast (i.e., data being exchanged confidentially only among the members in the group), a naive way is to employ the aforementioned secure unicast technique between every pair of group members or between the server and every client in the multicast group. Clearly this is not scal-

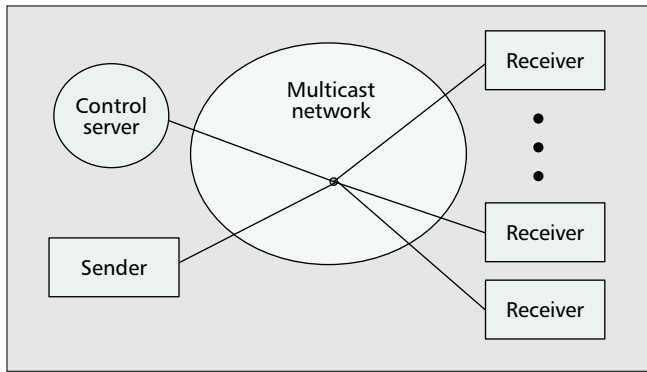
able: for N group members, $O(N^2)$ unicast connections must be established. One of the major challenges of offering secure multicast is to provide a secure service to a large group of users whose identities are generally not known to each other.

We show in Fig. 1 an architecture of a secure multicast system with a single sender and multiple receivers, where the sender transmits data to receivers via a multicast network. The control server is responsible for generating, storing, and distributing keys to both the sender and the receivers. A secure multicast system supported by key management generally offers the following security features.

Authentication: There are two types of authentication, depending on whether the sender or receiver is verified (one of the current techniques that supports user verification is digital signature [2]). Sender authentication is the process of reliably verifying the identity of the sender. Receiver authentication, on the other hand, aims at verifying the identity of the receiver. Since the sender usually provides the service, in some multicast applications (such as video services) only receiver authentication is required. One of the difficulties in offering authentication in multicast is how to efficiently validate the identities of a large number of users. If a group of users join a multicast group simultaneously, it may incur further problems such as join implosion.

Data Confidentiality: Confidentiality means that the content of a message must be shared only by authorized users. Unauthorized intruders should not be able to make sense out of the message by simply sniffing (or eavesdropping) on it. In unicast communication, it can be achieved by encrypting the data using, for example, the well-known RSA [3]. However, offering data confidentiality is more difficult in the multicast environment because a large number of users may be involved and membership may change quite rapidly.

This work was supported in part by the University Grant Council in Hong Kong, and by the Areas of Excellence (AoE) on Information Technology funded by the University Grant Council in Hong Kong (AoE/E-01/99), and by the Sino Software Research Institute (SSRI00/01.EG04) in the HKUST.



■ Figure 1. An example of a multicast system.

Integrity: Integrity means that if the data is altered in transit over the network, such an alteration should be detected. One method to achieve this is message digest (e.g., MD5) [4]. Efficient techniques for unicast communication may be straightforwardly applied in the multicast environment.

In addition to the above, other features need to be provided, such as traffic confidentiality (i.e., protection of traffic information such as its patterns from disclosure); non-repudiation (i.e., neither the sender nor the receiver of a message can deny the transmission); access control (i.e., access to information resources may be controlled by or for a system); and service assurance (i.e., resistance to certain attacks, such as denial of service). These features, however, are not related to key management issues and hence are out of the scope of this article.

Data confidentiality is one of the most challenging problems in secure multicast. To achieve this, a secure multicast scheme must address key management issues, which include efficient organization and distribution of keys with low communication overheads, key storage cost, and scheme complexity. This article reviews a number of such issues. Given that group membership is dynamic, most of this article focuses on reducing key delivery overheads and key storage requirement in this environment.

Current key management mechanisms can be roughly categorized into four groups: key tree-based approaches; contributory key agreements supported by the Diffie-Hellman algorithm; computational number theoretic approaches; and secure multicast framework approaches. After reviewing some of the security issues and performance criteria for secure multicast, we introduce the characteristics of each category and illustrate how they work via examples. We end with some comparisons and then with some concluding remarks.

Security Issues and Performance Criteria

Due to the “open” nature of multicasting (i.e., no membership control in the current protocols such as DVMRP, CBT, and PIM-DM [5–7]), users in the network can join and leave a multicast group at any time.¹ Because of this, one can easily eavesdrop data sent via multicasting. Data confidentiality can be achieved by encrypting the multicast data with a key, whereby the corresponding decryption key (the so called “group key”) is shared among every current group member. The main challenge is to allow only those users who have access rights to the multicast group to have the group key. In

¹ Some new multicast protocols extend the traditional routing algorithm to remedy this. However, there are still other security and practicality issues that need to be addressed [8].

Symbol	Definition
	Concatenation
$\{message\}_{key}$	Message encrypted with the key “key”

■ Table 1. Table of notation.

other words, a new member should not be able to decrypt the multicast data sent *before* his joining (the so-called “backward secrecy”) and a former member should not be able to decrypt the multicast data sent *after* his leaving or eviction (the so called “forward secrecy”) [9].

In order to provide such secrecy, whenever there is a membership change the data must be re-encrypted using a different key, and the corresponding decryption key (i.e., the group key) must be made known to all the members in the group. The “Re-key” message is used to notify all members of any key change, and the new key information (such as whether it is an actual key or the information for generating a new key). Since there may be a large number of users in a multicast group, such overheads in key distribution and update can be high if not managed properly.

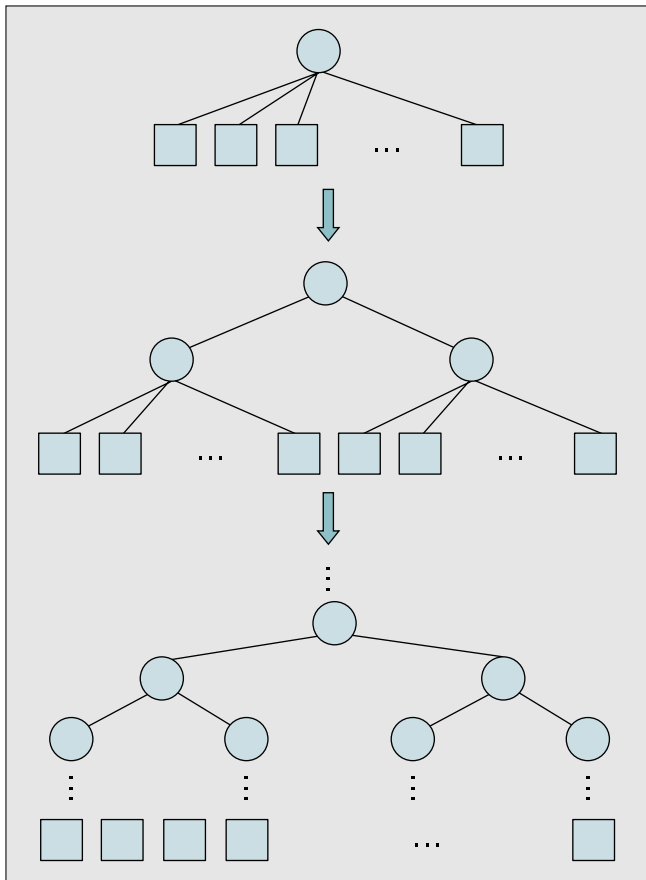
There are several criteria for evaluating a key management scheme. Note that different applications may focus on different sets of criteria, depending on their specific concerns. For example, a satellite pay TV system may be concerned more with key storage and transmission requirements, while a military system may be concerned more with security criteria. We list below some important performance criteria. Note that we have used | to denote concatenation (e.g., $m_1|m_2$ signifies that message m_1 concatenates with message m_2), and $\{message\}_{key}$ to denote encryption (e.g., $\{m\}k$ signifies that message m is encrypted with key k) (Table 1).

Scalability: Scalability refers to the handling of key changes and efficient data distribution to a large group so as to reduce key storage in both the control servers and users. An efficient secure multicast system should be able to deal with a large and highly dynamic multicast group. The following criteria are usually used to evaluate the scalability of the system.

- The number of keys stored in each member and control server: In order to decrypt the multicast data, each member must receive (or generate) the group key and store it locally. In order to communicate securely with the control server, each member may have an individual private key (if asymmetric keys are used). Besides these two keys, some key management schemes introduce yet another type of key termed an “auxiliary key” (a.k.a., subgroup key or key-encrypted key), which is used to reduce the total number of re-key messages. All these keys incur storage overheads. A control server must store all the keys in use. Clearly, a large key pool not only requires extensive memory but also increases the complexity of key management.
- The number (or size) of re-key messages: All group key and auxiliary keys have to be updated upon membership change and sent to the corresponding members. If re-key messages are not managed properly, they may incur other overheads, such as key re-transmission or key synchronization.
- The number of keys generated on membership change: In order to reduce the key storage requirement or the number of re-key messages, some keys are generated in the control server or by users upon membership change. Such a burst of key generation may be processing-intensive. This is especially critical for a low-power device.

Security: Some security criteria are listed as follows.

- Strength against collusion: Colluders are non-members of a multicast group but they collude with each other to



■ Figure 2. Extension of star structure to multi-level tree structure to reduce re-key overheads.

retrieve the multicast data sent to the group. Some colluders may have certain knowledge of the multicast group. If the aggregated knowledge satisfies some certain conditions, they can decipher the encrypted data. In a collusion-vulnerable system, there is generally a trade-off between key transmission or key storage and the resistance against collusion.

- **Knowledge of the group members:** Group members may be required to be involved in key distribution or generation. This can be a potential security loophole when members attack the system using their prior knowledge of key generation or key distribution. Therefore, it is generally advisable that members should not know too much about key generation, key distribution, and the mechanisms involved in establishing the security of the system.

Other Criteria: Besides the above, we list below other criteria for secure multicast:

- **The cost of establishing a secure multicast system:** There are two kinds of costs: the cost of establishing a control manager and the infrastructure of the system, and the cost of the user machines.

- **Implosion avoidance:** Implosion generally occurs at the point to which users are directed (such as the point to receive key information). A centralized control server may also have an implosion problem when a burst of members joins or leaves the system.

- **Symmetric key versus asymmetric key:** Since a symmetric key is easier to generate and encryption/decryption algorithms based on it is less processing-intensive, a system based on symmetric keys in general performs faster and is more applicable to handheld devices such as smart cards. However, secret sharing becomes critical in this system, i.e., trusted parties must be involved in this system.

- **Reliable multicast:** Unlike TCP, traditional IP multicast is based on best-effort delivery. Since some schemes use multicast to distribute keys, keys may be missed, which leads to overheads such as key synchronization. Therefore, reliable multicast is required for these schemes.

Key Tree-Based Approach

In order to motivate the key tree-based schemes, we first discuss two simple (non-scalable) solutions for key management.

In the first solution each member establishes a pairwise unicast channel with the control server. When a member joins or leaves the multicast group, the server generates and distributes a new group key to each of the current members.

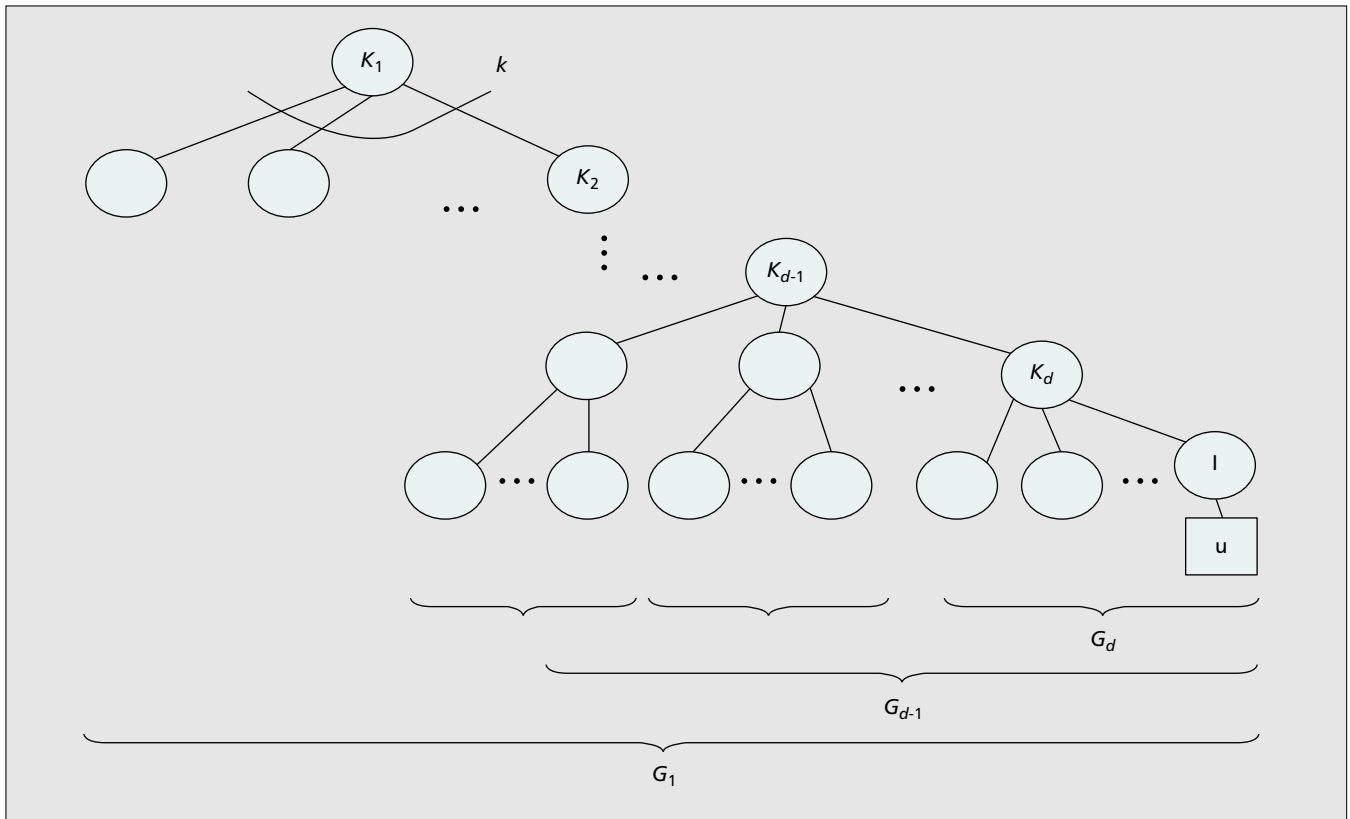
In this scheme, only two re-key messages are sent when a new member joins: the new group key is encrypted by the old group key and sent via multicast; and the new group key is encrypted by the new member's individual key and sent via unicast to the new member. However, the cost for a group key update upon a user's leaving is proportional to the size of the group. Clearly, this is a "join-friendly" approach.

In the second solution a key is assigned to every possible subset of the current members. Therefore, each member holds a number of keys equal to the total number of possible subsets that include the member. In other words, for a group size of N members, the server and each member must hold $O(2^N)$ keys, which is high for a large N . Whenever a member leaves the group, the multicast data can be encrypted with the key corresponding to the subset without the departed member. However, when a new member joins the group, new keys must be generated and transmitted to every possible subset of members formed with the new member. Clearly, unlike the former scheme, this scheme is "leave-friendly" because a user's leaving does not incur any overheads, but there are high overheads for a user's joining.

Let us illustrate the above by an example. Say there are three members of a group, A , B , and C . The server then holds the following keys: K_{ABC} , K_{AB} , K_{BC} , K_{AC} , and the individual keys, while A holds the decryption keys corresponding to K_{ABC} , K_{AB} , and K_{AC} , and likewise for B and C . The data is encrypted with K_{ABC} in order to be multicast to all of the members, and the members use the corresponding decryption key to decrypt the data. If a member, A for example, leaves the group, the server simply uses K_{BC} to send the data to B and C . Clearly, no new key needs to be generated for a member's leaving. However, if a new member, D for example, joins the group, then new keys — K_{ABCD} , K_{ABD} , K_{BCD} , K_{ACD} , K_{AD} , K_{BD} , K_{CD} — must be generated and sent to the relevant members.

Clearly, both schemes are not scalable in terms of the number of users and dynamic traffic with members frequently joining and leaving. Therefore, a tree structure for arranging keys has been proposed, and we illustrate here how this benefits the re-key messaging. In Fig. 2, the topmost diagram shows a star structure used for the first simple (non-scalable) solution, where the circled nodes represent keys and squared nodes represent members. Since all members share only the same group key (the root node), whenever there is a membership change, the new group key has to be distributed individually to each member.

To increase the scalability of the system, we split the members into two (or more) groups and create a hierarchy with two levels (refer to the second diagram in Fig. 2). Now, each member belongs to one of the two subgroups and shares the same subgroup key (i.e., the second-level internal node). Therefore, if there is a membership change in a subgroup, the subgroup key can only be distributed to the corresponding



■ Figure 3. A k -ary key tree.

subgroup members instead of the entire group, thereby cutting half of the key transmission overhead. The group key can then be sent to all members using the subgroup keys instead of sending it to each individual member. To reduce the re-key overhead within a subgroup, we can “recursively” split each subgroup to form multi-level subgroups. In this way, we extend the hierarchy of two levels to a tree structure. By arranging the keys in such a way, the total number of re-key messages sent by the control server can be greatly reduced. There are a few schemes based on this key tree-based approach [10–14].

The key tree-based approach arranges all group members at the leaves of a “logical” key tree stored in a control server. Due to the hierarchical nature of the tree, different members may share the same nodes with other members (e.g., the root is shared by all members). Associated with each node is a key. All the members emanating from the same node share a common key, the so called “subgroup key.” The key at the root is the group key. In this way, a member is associated with the other members through sharing the same key. Note that these subgroups are not mutually exclusive so a member can belong to more than one subgroup. Whenever a subgroup of members needs to be informed that there is a key change, we can encrypt the new key with the subgroup key and send it via multicast instead of unicasting to each of the subgroup members.

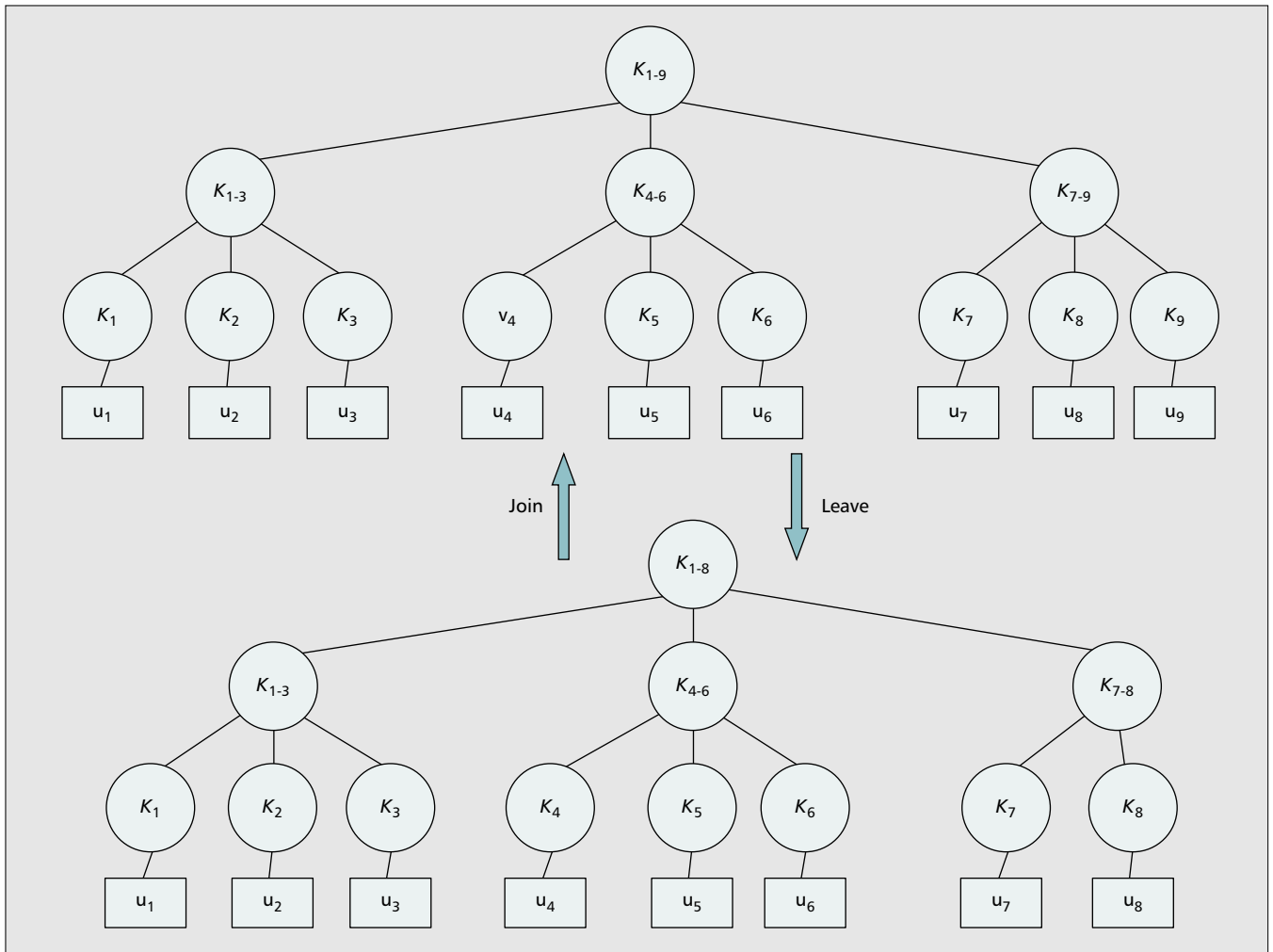
Therefore, each member is assigned the same number of keys as the height of the key tree. The number of re-key messages is hence logarithmic to the group size according to the tree height. However, because of the many keys in the system, higher storage requirements for both the control server and members is necessary. Therefore, there is a trade-off between key transmission and key storage [15].

We now discuss more specifically the hierarchical key tree scheme (a.k.a. logical key hierarchy) [10, 11]. The hierarchical key tree is a logical tree structure of a multicast group. The

tree is stored in the control server. In the tree, group members are arranged at the leaves and the internal nodes store keys (see Fig. 3 for a k -ary tree with depth d). There are three types of keys. The first type is a group key, K_1 , used to encrypt/decrypt multicast data; the second type is a subgroup key (such as K_{d-1} and K_d) used to encrypt/decrypt other keys instead of the actual data; the last type is the individual key, l . Each member holds the keys along the path from itself (the leaf) all the way to the root. Therefore, for the case of member u , u holds K_1, \dots, K_{d-1}, K_d . Each subtree in the entire key tree is a subgroup and each member is assigned to more than one subgroup. For example, member u belongs to groups G_d, G_{d-1}, \dots, G_1 .

When a membership change occurs, apart from the group key, all keys held by the new or former member must be changed in a bottom-up manner. For example, if u leaves the group, we first need to change K_d to a new subgroup key, say K'_d , and send it to all the members who shared K_d with u (i.e., u 's siblings in the tree). Since K_d is known by u , the control server has to encrypt K'_d by each members' individual key and send it to them by unicast. After sending K'_d , the process can be propagated one level up. Now, K_{d-1} must be changed. Since K_d is changed to K'_d , which is unknown to u , the control server can encrypt the new K_{d-1} by all subgroup keys, including K'_d , and send it to all subgroups in the $(d - 1)$ th level. We repeat the same process upward one level at a time until it reaches the root where K_1 is changed. Then all the keys, including the group key, held by u are changed.

If u is a new member joining the group, in order to guarantee backward secrecy, all the keys from K_d to K_1 must be changed. Since u knows nothing about the keys in the group, when the control server changes K_d to a new key K'_d , then K'_d can be encrypted by K_d and multicast to u 's sibling and unicast to u . Similarly, this process can be propagated upward one level at a time, with the control server multicasting the new keys to the subgroups under the key and unicasting the key to u .



■ Figure 4. An example of ternary hierarchical key tree.

If we assume that the key tree is a k -ary full tree, after each membership change, the number of re-key messages per leave and join are proportional to the depth of the key tree, $\log_k n$, where n is the group size. For each leaf, the updated keys at each level must be sent k times (one for each branch). For each join, the updated keys at each level must be sent twice (one for multicasting to the old members and one for unicasting to the new member). Therefore, the number of re-key messages per leave and join are $k \log_k n$ and $2 \log_k n$, respectively.

As an example, in Fig. 4 we show a hierarchical key tree of degree three with nine members, u_1 to u_9 . The key at the root, K_{1-9} , is the group key. K_{1-3} , K_{4-6} , and K_{7-9} are subgroup keys for subgroups $\{u_1, u_2, u_3\}$, $\{u_4, u_5, u_6\}$, and $\{u_7, u_8, u_9\}$, respectively, and K_1 to K_9 are individual keys for each member.

Suppose that u_9 leaves the group, and hence the remaining eight members, $u_1 - u_8$, form a new secure group. A new group key, K_{1-8} , is then generated, and u_7 and u_8 now form a new subgroup that requires a new subgroup key, K_{7-8} . In order to send K_{7-8} to u_7 and u_8 , the control manager encrypts it with K_7 before sending it to u_7 and with K_8 before sending it to u_8 . Finally, the control manager sends the new group key K_{1-8} to the members in each subgroup by encrypting the group key with each subgroup's key.

On the other hand, if u_9 is a new member joining the group, K_{1-8} and K_{7-8} must be changed to K_{1-9} and K_{7-9} , respectively. Since u_9 knows nothing about the keys in the group before it joins, K_{7-9} can be sent to u_7 and u_8 by encrypting it with the previous subgroup key, K_{7-8} , while the new

group key, K_{1-9} , can be encrypted with the previous group key, K_{1-8} , before multicasting it to members u_1 to u_8 . Finally, the control manager sends K_{1-9} and K_{7-9} to u_9 by unicast.

Obviously, a hierarchical key tree scheme reduces the number of re-key messages to $O(\log n)$. Since a user belongs to multiple multicast groups, it increases the difficulties of a control server in managing all group members. Indeed, given a pool of users, it has been shown that there is an optimal group size to achieve minimum transmission overhead [16, 17]. In addition, since subgroup keys are sent via multicast, reliable multicast is required.

In order to further reduce the number of re-key messages, researchers have proposed other key tree schemes that make use of a one-way function to generate a sequence of keys from the lower subgroups to the upper levels [12, 14]. For instance, in [12] a one-way function key generator is applied on the user's side to generate the subgroup keys along the path from itself to the root. Since some keys are generated on the user's side, these schemes can further reduce the number of re-key messages. However, they increase the workload of the user.

Contributory Key Agreement Supported by the Diffie-Hellman Algorithm

Some key-management schemes extend the well known Diffie-Hellman key-exchange algorithm [18] to support group key agreement [19–23]. Instead of the control server generating and distributing a group key to all members, these schemes

focus on the issue of “key agreement.” In contributory key-agreement schemes, generating and distributing a group key is the responsibility not only of the control server, but also every member.

The two-party Diffie-Hellman algorithm provides neither encryption nor signatures but allows two individuals to agree on a common (symmetric) key. After agreeing on a shared key, the individuals can send messages to each other securely. In this system, a common generator, g , and a common prime number, p , are required. The generator accepts a secret number, S_i , randomly generated by member i and outputs a partial key (i.e., the key information to generate the group key), g^{S_i} ($= g^{S_i} \text{ mod } p$). If two individuals, M_1 and M_2 for example, agree on a shared key, they first exchange their partial keys (i.e., g^{S_1} and g^{S_2}). Afterward, they can compute the shared key, $g^{S_1 S_2}$. It has been proved that nobody else can compute the shared key $g^{S_1 S_2}$ in a reasonable amount of time even though they know g^{S_1} and g^{S_2} .

All the contributory key-agreement schemes supported by the Diffie-Hellman algorithm are based on computing a subset of $\{g^{\Pi(A)} \mid A \subset \{S_1, \dots, S_n\}\}$ from $g^{\Pi_{j=1, j \neq i}^n S_j}$, member M_i can easily compute the shared group key $g^{\Pi_{j=1}^n S_j}$.

In [22] a scheme called CLIQUES is proposed to handle contributory key agreement. Unlike key tree-based approaches, CLIQUES arranges the group members in a logical linear structure and passes key information sequentially. The group members are indexed. The last two members (the two highest indexed members) are responsible for taking part in some special steps of key distribution, with the last member also a control server. The process of key initialization of CLIQUES can be divided into four stages as follows:

1) Each member M_i receives a partial key $g^{\Pi\{S_j \mid j \in [1, i-1]\}}$ from his previous member, M_{i-1} . It then computes a new partial key $g^{\Pi\{S_j \mid j \in [1, i]\}}$ by adding his own secret number. This partial key is passed sequentially until it reaches the second to last member, M_{n-1} , at which the first stage finishes.

2) Member M_{n-1} multicasts his partial key, $g^{\Pi\{S_j \mid j \in [1, n-1]\}}$, to all other members except M_n . After all the members receive the partial keys, each member factors out his own secret number by using his inverse function, S_i^{-1} . The partial key for M_i becomes $g^{\Pi\{S_j \mid j \in [1, n-1] \wedge (j \neq i)\}}$.

3) All the members (except M_n) are required to send their factorized partial keys to M_n (M_n has the responsibility to store these factorized partial keys for future use).

4) M_n puts his secret number into all the factorized partial keys, $\{g^{\Pi\{S_j \mid j \in [1, n] \wedge (j \neq i)\}} \mid i \in [1, n-1]\}$, and sends them back to the corresponding members. Each member computes the group key by combining the partial keys he receives and his own secret number.

To offer backward secrecy, whenever a new member joins the multicast group, the new member M_{n+1} replaces M_n to distribute partial keys in stage four. First, M_n factorizes out his secret number from all these factorized partial keys and then adds a newly generated secret number, S'_n . The new keys become $\{g^{S'_n \Pi\{S_j \mid j \in [1, n-1] \wedge (j \neq i)\}} \mid i \in [1, n]\}$, which M_n sends to M_{n+1} . After M_{n+1} receives all these keys, he adds his own secret number (as in stage 4) and sends the new partial keys, $\{g^{S'_n \Pi\{S_j \mid j \in [1, n+1] \wedge (j \neq i) \wedge (j \neq n)\}} \mid i \in [1, n+1]\}$, back to the corresponding members.

Offering forward secrecy is relatively easy. When a member, M_l for example, leaves the group, M_n sends the new par-

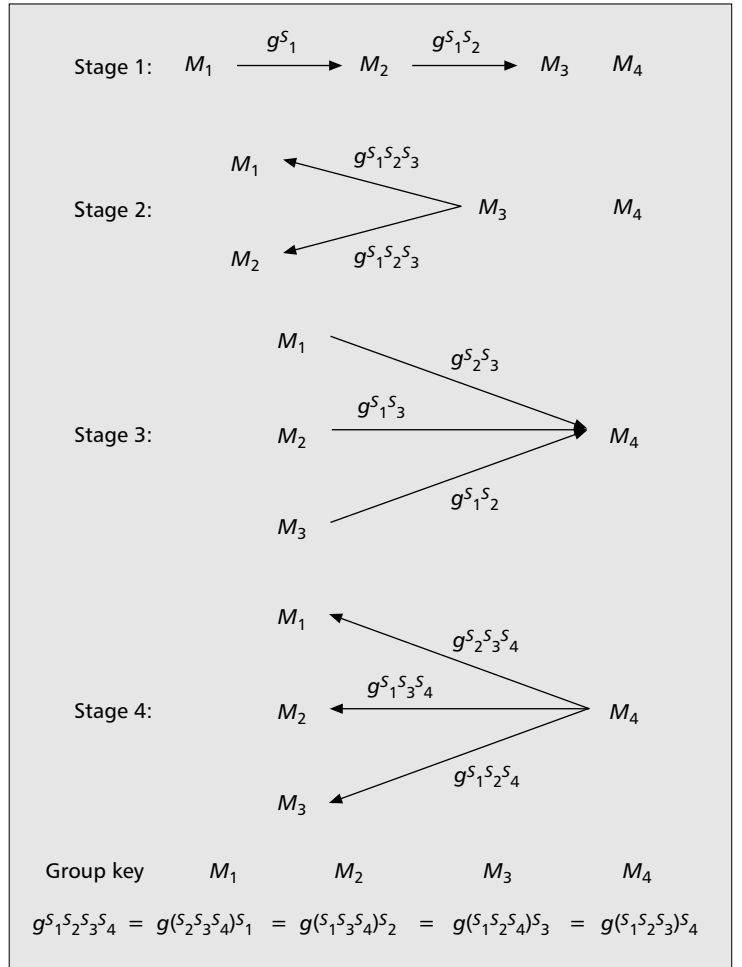


Figure 5. An example of key initialization.

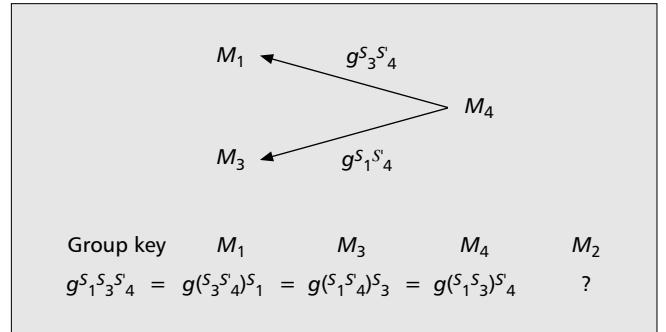


Figure 6. A re-key example for a member leaving a multicast group.

tial keys excluding the secret number of M_l , $\{g^{\Pi\{S_j \mid j \in [1, n] \wedge (j \neq i) \wedge (j \neq l)\}} \mid i \in [1, n] \wedge (i \neq l)\}$, to all the members besides M_l . Since the new group key becomes $g^{\Pi_{j=1 \wedge j \neq l}^n S_j}$, M_l has no information to compute the new group key. Therefore, M_l is successfully evicted from the group communication.

We show an example of key initialization and re-keying in Figs. 5, 6, and 7. In Fig. 5, there are four members, M_1 to M_4 , agreeing on a shared group key according to the initialization steps mentioned above.

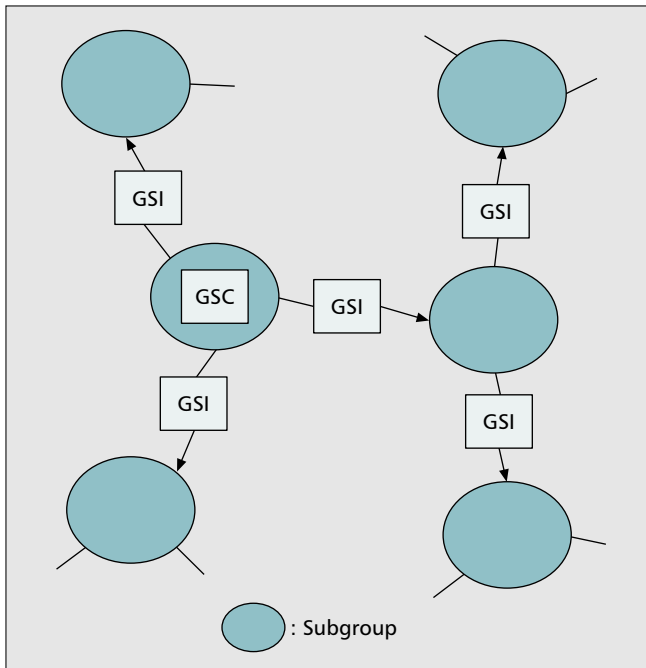
Figure 6 shows a re-key example for a member leaving a multicast group. If M_2 leaves the group, M_4 only needs to send

the new partial keys with his new secret number, S'_4 , to the corresponding members. Then all current members compute the new group key, $g^{S_1 S_3 S'_4}$, but the former member, M_2 , does not have enough information to compute the new group key.

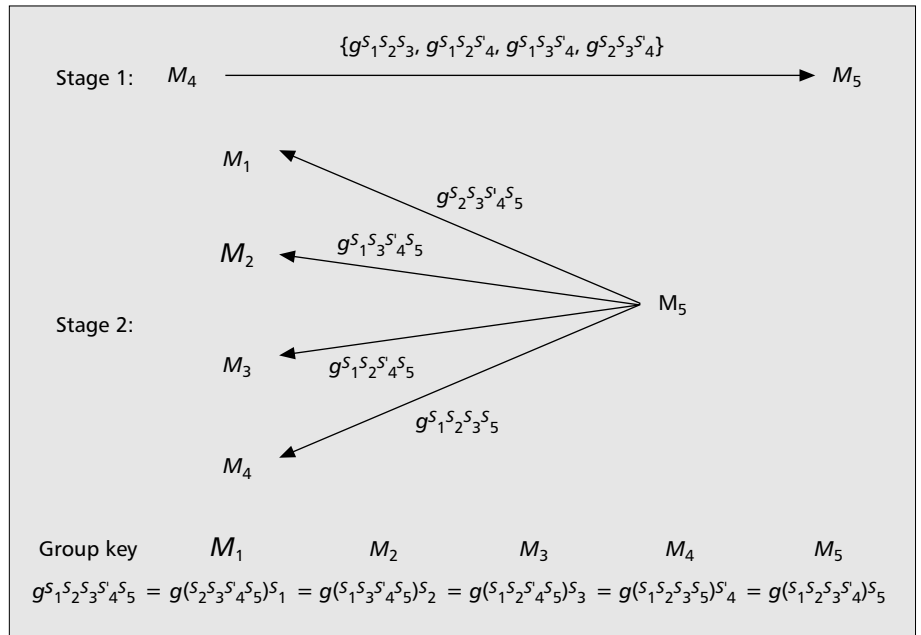
Figure 7 shows a re-key example for a member joining a multicast group. When a new member M_5 joins the group, first M_4 sends the partial keys with his new secret key, S'_4 , to M_5 and then M_5 sends the new partial keys with his secret number, S_5 , to the corresponding members. Finally, all the members collect all the necessary partial keys for computing the new group key, $g^{S_1 S_2 S_3 S'_4 S_5}$.

In this scheme the control server (i.e., the last group member) is only responsible for storing the partial key from all the members. Since it does not generate any key, it can be a less powerful device. Furthermore, since a symmetric key is used, all members can use a low-end device such as an inexpensive set-top box to encrypt and decrypt multicast data. This scheme, however, suffers from the scalability problem, i.e., the number of re-key messages and the number of partial keys exchanged in the initialization stage are proportional to the multicast group size. Furthermore, since the control server and the newly joined member must collect the partial keys from other members, implosion problems may arise.

Instead of organizing group members in a linear manner, other schemes organize members into other structures such as tree and d-dimensional hypercube [23, 24]. The schemes based on different structures can efficiently reduce the number of partial keys in the initialization stage to $O(\log n)$.



■ Figure 8. Architecture of the Iolus framework.



■ Figure 7. A re-key example for a member joining a multicast group.

Computational Number Theoretic Approach

The computational number theoretic approach allows group members to compute a shared group key according to some key information sent from other members or the control server. The Diffie-Hellman cryptography algorithm mentioned in the previous section is actually a computational number theoretic approach in which each member “contributes” some of its own information (i.e., its partial key) in order to agree upon a new common key for data encryption. The schemes introduced in this section do not have such a contributory characteristic, i.e., the members do not exchange information among themselves in order to generate a new group key. A number of schemes based on the computational number theoretic approach have been proposed previously [25–30].

In [25] a scheme named “secure lock” is proposed. In this scheme, when a member joins or leaves the group, the group members receive a re-key message that can be used to generate a shared group key based on the Chinese Remainder Theorem (CRT).

CRT states that if there are n pairwise relatively-prime numbers, N_1, N_2, \dots, N_n , and another set of positive integers, R_1, R_2, \dots, R_n , then we can form a set of congruous equations, $X \equiv R_i \pmod{N_i}, \forall i \in [1, n]$, that have a common solution X . In other words,

$$X = \left(\sum_{i=1}^n \frac{\prod_{j=1}^n N_j}{N_i} R_i f_i \right) \pmod{\left(\prod_{j=1}^n N_j \right)},$$

where f_i is a positive integer obtained by solving the following equation for all i ,

$$1 \equiv \left(f_i \frac{\prod_{j=1}^n N_j}{N_i} \right) \pmod{N_i}.$$

To apply the CRT to key management, the common solution of the congruous equations can be used to generate a shared group key for all the group members. First, each member is assigned a positive integer, N_i for example, for member i . R_i represents the set of the group key encrypted with the public key of each member, i.e., if G denotes the group key

and k_i denotes the public key of member i , then R_i is equal to $\{G\}k_i$.

Whenever there is a membership change, the control server computes and sends a new common solution, X , according to the set of pairwise relative primes, N_i , and the new encrypted group keys, R_i s. After a member, M_i , receives X (i.e., the re-key message), M_i computes R_i from X . Since R_i is the group key encrypted with M_i 's public key, M_i can decrypt X with his private key to retrieve the new group key.

Secure lock makes good use of CRT to let each member compute the group key. However, it is not scalable because the size of the secure lock (re-key messages) and the time to generate the set of encrypted group keys are proportional to the group size. Therefore, this approach only works well in small systems.

Secure Multicast Framework

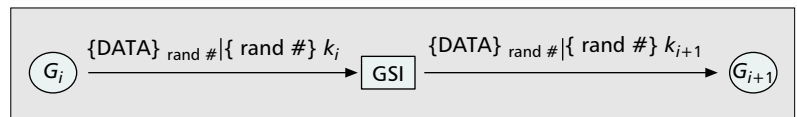
A secure multicast framework is a distributed system in which some intermediate participants such as trustable agents (with the agents in some cases integrated as group members) work as the control servers to deliver or generate keys. Users are divided into different physical groups which are served by their respective agents. In this way, any key change is done within a subgroup. We illustrate the framework by using as an example the Iolus framework [31]. (There are other schemes based on this secure multicast framework. Readers interested in them are referred to [32–34] and the references therein.)

The Iolus framework deals with the scalability issues of a multicast group by partitioning a large multicast group into subgroups and employing a hierarchy of group security agents to “relay” key and encrypt data. Therefore, join and leave requests can be handled within the respective subgroups. Similar to the idea of a key tree-based approach, Iolus divides members into a hierarchy of subgroups. However, members are arranged into physical subgroups as opposed to a logical hierarchical key tree.

Figure 8 shows the architecture of the Iolus framework, which consists of a number of group security agents (GSAs). The GSA responsible for controlling the top-level subgroup (root) of the hierarchy is called the group security controller (GSC), while other GSAs are called group security intermediaries (GSIs). GSIs sit between subgroups and are responsible for managing their own subgroups.

A GSI works as a relay. It decrypts the data sent from the upper-level subgroup and multicasts the data to its downstream subgroup after re-encrypting the data with its own subgroup key. To reduce the overhead of re-encryption for the entire multicast data, the sender encrypts the data using a random key, which is in turn encrypted with the GSC key and concatenated to the actual data. For each “key translation,” only the random key needs to be re-encrypted by GSI. For example, in Fig. 9 the GSI between subgroup G_i and G_{i+1} , upon receiving data from G_i , decrypts the random key and then re-encrypts the data using its subgroup key, $\{rand\# \}k_{i+1}$, before forwarding it to the members in G_{i+1} .

Regarding the join operation, a new member first locates its subgroup's GSA before sending a join request to this GSA. To provide backward secrecy, the GSA generates a new subgroup key and sends it to its subgroup members (including the other GSIs attaching to the subgroup). The new subgroup key can be encrypted with the old subgroup key before multicasting it to its subgroup members and encrypted with the new member's individual key before sending it to the new member. For leave request, a new subgroup key must be generated by its GSA and distributed to all the current subgroup members



■ Figure 9. Key translation in a GSI.

(including the attaching GSIs). Since the old subgroup key is known to the former member, it cannot be used to encrypt the new key. Therefore, the new subgroup key must be encrypted by each member's individual key and sent to the corresponding members via unicast channels.

The Iolus framework breaks a large and wide-area multicast group into smaller subgroups, leading to easy management. As each subgroup is likely to be an autonomous system (AS), the GSA can be a control server of the AS and perform some administrative work (such as authentication). However, the Iolus framework suffers from several drawbacks. Since there is a GSA in every subgroup, the system cost increases. Furthermore, since the GSA must perform key translation, it introduces message delay, especially when the group is large or when the GSAs are not powerful enough. Therefore, the framework is not very suitable for some real-time multicast applications such as video-conferencing.

Comparison

According to the performance criteria listed earlier, we present the comparison among different key-management schemes in Table 2. We have used the following nomenclature: n is the group size, k is the branching factor of a key tree, m is the number of subgroups, and d_i is the distance from GSA to the subgroup to which member i belongs. In order to compare the schemes, we have made the following assumptions:

- In the case of a hierarchical key tree, the tree is balanced and full.
- In CLIQUES and secure lock, a partial key and a secure lock are involved as the re-key message.
- In the Iolus secure framework, members are evenly distributed in each subgroup so that the number of members in each group is n/m .

We clearly see that different approaches have their strengths and weaknesses. There is not a single scheme that has an overriding advantage over the others. In reality, a scheme should therefore be chosen depending on the particular application requirements.

Conclusion

Many multicast applications require data confidentiality. Key management, which involves key distribution (re-key messaging), key storage, and key generation, is the main issue of data confidentiality. In this article, we have discussed this issue and some performance criteria for evaluating a key-management scheme for multicast applications. We have reviewed the techniques of some of the current key-management schemes, which can be roughly divided into four categories:

- Key tree-based approaches, in which a tree is used to divide members into subgroups so as to reduce the number of re-key messages.
- Contributory key agreement methods supported by the Diffie-Hellman algorithm, which extends the two-party Diffie-Hellman algorithm to support multicast security and allows members to “agree” upon a symmetric group key.
- Computational number theoretic approaches, which use number theory so that members can compute a shared group key according to the information sent from a control server.

- A secure multicast framework, which introduces agents to share the workload of a control server for key management, and assigns members to different subgroups so as to reduce re-key messaging within a subgroup.

In each category, we have illustrated the fundamental mechanisms and addressed their strengths and weakness. We also compared them in terms of the number of keys stored by the server and members, re-key messages for leave and join, complexity, and so on.

References

- [1] S. E. Deering, "Multicast Routing in Internetworks and Extended LANs," *Proc. ACM SIGCOMM*, vol. 18, Aug. 1998, pp. 55–64.
- [2] NIST, "NIST: FIPS 186 for Digital Signature Standard," NIST: FIPS 186 for Digital Signature Standard (DSS), May 1994.
- [3] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Commun. ACM*, Feb. 1978 pp. 120–26.
- [4] R. L. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, Apr. 1992.
- [5] S. E. Deering, "Multicast Routing in a Datagram Internetwork," Ph.D. thesis, Stanford University, Palo Alto, California, Dec. 1991.

	Hierarchical key tree	CLIQUEs	Secure lock	Iolus
Number (or size) of keys stored in a control manager	$O(n)$	$O(n)$	$O(n)$	$O(n/m)$ for a linear structure; $O(n/m + m)$ for a star structure
Number (or size) of keys stored in each member	$O(\log n)$	$O(1)$ for a system with a static control manager; $O(n)$ for a system with a dynamic control manager	$O(1)$	$O(1)$
Number (or size) of re-key messages distributed by a control manager (or an agent) on a join	$O(\log n)$	$O(n)$	$O(n)$	$O(1)$
Number (or size) of re-key messages distributed by a control manager (and agents) on a leave	$O(k \log n)$	$O(n)$	$O(n)$	$O(n + m)$ in total; $O(n/m)$ per each control manager or agent for the best case while $O(n/m + m)$ for the worst case
Number (or size) of re-key messages received by the new member on a join	$O(\log n)$	$O(1)$	$O(n)$	$O(1)$
Number (or size) of re-key messages received by each old/remaining member on a join/leave	$O(1)$ for the best case; $O(\log n)$ for the worst case	$O(1)$	$O(n)$	$O(1)$
Processing time for retrieving a group key	$O(\log n)$ (caused by decrypting a burst of subgroup keys)	$O(1)$	$O(n)$ (since size of secure lock is proportional to the group size)	$O(d_i)$ (caused by key translation): $O(1)$ for the best case; $O(m)$ for the worst case
Cost of establishing a control manager and the infrastructure of the system	Medium	Low	Medium	High
Powerfulness of a user machine	Medium	Low	Low	Low
Vulnerable to implosion	No	Yes (caused by initialization process)	No	No
Symmetric or asymmetric group key?	Asymmetric key	Symmetric key	Asymmetric key	Asymmetric key
Single point of failure	Yes	No (assume dynamic control manager is used)	Yes	Yes
Reliable multicast required?	Yes	No	Yes	Yes
Applications	A general multicast system with a mildly large group of members such as Internet radio and stock quote service.	A multicast system with a small group of members and a less powerful server (or without a centralized server) such as video conference.	A multicast system with a small group of members	A general multicast system with widely (or globally) distributed members such as pay-per-view international news and movie system.

■ Table 2. A comparison of key management schemes (n is the group size, k is the branching factor of a key tree, m is the number of subgroups, and d_i is the distance from GSA to the subgroup to which member i belongs.)

- [6] A. Ballardie, P. Francis, and J. Crowcroft, "Core-Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing," *Proc. ACM SIGCOMM*, Oct. 1993, pp. 85–95.
- [7] S. E. Deering *et al.*, "An Architecture for Wide-Area Multicast Routing," *Proc. ACM SIGCOMM*, (London, UK), Sept. 1994, pp. 126–35.
- [8] C. Shields and J. J. Garcia-Luna-Aceves, "KHIP: A Scalable Protocol for Secure Multicast Routing," *Proc. SIGCOMM*, Sept. 1999, pp. 53–64.
- [9] W. Diffie, "Authenticated Key Exchange and Secure Interactive Communication," *Proc. SECURICOM*, (Paris, France), 1990 pp. 399–406.
- [10] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key Management for Multicast: Issues and Architectures," RFC 2627, June 1999.
- [11] C. K. Wong, M. Gouda, and S. S. Lam, "Secure Group Communication Using Key Graphs," *IEEE/ACM Trans. Net.*, vol. 8, no. 1, Feb. 2000, pp. 16–29.
- [12] R. Canetti *et al.*, "Multicast Security: A Taxonomy and Some Efficient Constructions," *Proc. INFOCOM '99*, 1999, vol. 2, pp. 708–16.
- [13] I. Chang *et al.*, "Key Management for Secure Internet Multicast using Boolean Function Minimization Techniques," *Proc. IEEE INFOCOM '99, Conf. Comp. Commun.*, vol. 2, 1999, pp. 689–98.
- [14] D. Balenson, D. A. McGrew, and A. T. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees and Amortized Initialization," Internet Draft, 1999.
- [15] R. Canetti, T. Malkin, and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption," *Proc. Advances in Cryptology — EUROCRYPT '99 (Int'l. Conf. Theory and Applications of Cryptographic Techniques)*, 1999, pp. 459–74.
- [16] K.-C. Chan and S.-H. G. Chan, "Distributed Servers Approach for Large-Scale Secure Multicast," *IEEE JSAC*, special issue on network support for multicast communications, vol. 20, no. 8, Oct. 2002.
- [17] K.-C. Chan and S.-H. G. Chan, "Distributed Servers Networks for Secure Multicast," *Proc. IEEE Globecom '01*, (San Antonio, TX), 25–29 Nov. 2001.
- [18] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Info. Theory*, vol. IT-22, Nov. 1976, pp. 644–54.
- [19] M. Just and S. Vaudenay, "Authenticated Multi-Party Key Agreement," *Proc. Advances in Cryptology — ASIACRYPT: Int'l. Conf. Theory and Application of Cryptology and Info. Security*, LNCS, Springer-Verlag, 1996, pp. 36–49.
- [20] M. Burmester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System," *Proc. Advances in Cryptology — EUROCRYPT '94, Wksp. Theory and Application of Cryptographic Techniques*, Springer-Verlag, 1995, pp. 275–86.
- [21] G. Ateniese, M. Steiner, and G. Tsudik, "New Multi-Party Authentication Services and Key Agreement Protocols," *IEEE JSAC*, vol. 18, no. 4, Apr. 2000, pp. 628–39.
- [22] M. Steiner, G. Tsudik, and M. Waidner, "Key Agreement in Dynamic Peer Groups," *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 8, Aug. 2000, pp. 769–80.
- [23] Y. Kim, A. Perrig, and G. Tsudik, "Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups," *Proc. ACM Conf. Comp. and Commun. Security*, Nov. 2000, pp. 235–44.
- [24] K. Becker and U. Wille, "Communication Complexity of Group Key Distribution," *Proc. ACM Conf. Comp. and Commun. Security*, (New York, NY), 1998, pp. 1–6.
- [25] G. H. Chiou and W. T. Chen, "Secure Broadcasting Using the Secure Lock," *IEEE Trans. Software Eng.*, vol. 15, no. 8, IEEE Computer Society, 1989, pp. 929–34.
- [26] A. Fiat and M. Naor, "Broadcast Encryption," *Proc. Advances in Cryptology — CRYPTO '93*, Springer-Verlag, Berlin, Germany, 1994, pp. 480–91.
- [27] C. Blundo *et al.*, "Perfectly Secure Key Distribution for Dynamic Conferences," *Info. and Computation*, vol. 146, no. 1, Oct. 1998, pp. 1–23.
- [28] D. R. Stinson, "On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption," *Designs, Codes and Cryptography*, vol. 12, no. 3, 1997, pp. 215–43.
- [29] B. Chor *et al.*, "Tracing Traitors," *IEEE Trans. Info Theory*, May 2000, vol. 46, no. 3, pp. 893–910.
- [30] M. Abdalla, Y. Shavitt, and A. Wool, "Key Management for Restricted Multicast Using Broadcast Encryption," *IEEE/ACM Trans. Net.*, Aug. 2000, vol. 8, no. 4, pp. 443–54.
- [31] S. Mitra, "Iolus: A framework for Scalable Secure Multicasting," *Proc. ACM SIGCOMM'97*, 1997, pp. 277–88.
- [32] T. Hardjono, B. Cain, and I. Monga, "Intra-Domain Group Key Management Protocol," Internet Draft, IETF, 1998.
- [33] T. Hardjono, B. Cain, and N. Dorawamy, "A Framework for Group Key Management for Multicast Security," Internet Draft, IETF, 1999.
- [34] M. Waldvogel *et al.*, "The VersaKey Framework: Versatile Group Key Management," *IEEE JSAC*, vol. 17, no. 9, Sept. 1999, pp. 1614–31.

Biographies

KIN-CHING CHAN (ching@cs.ust.hk) received the B.Sc. and M.Phil. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 1999 and 2001, respectively. He is currently with Roctec Technology Ltd., developing networking solutions for the company. In 1998–99, he was a visiting student at the Computer Science Department, University of Minnesota at Twin Cities, MN. His research interests include multimedia networking, multicast protocols, computer security, cryptography, and coding theory.

S.-H. GARY CHAN [M'98] (gchan@cs.ust.hk) received the Ph.D. degree in electrical engineering with a minor in business administration from Stanford University, Stanford, CA, in 1999, and the B.S.E. degree (highest honor) in electrical engineering from Princeton University, Princeton, NJ, in 1993. He is currently an assistant professor with the Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, and an adjunct researcher with Microsoft Research Asia in Beijing. He was a visiting assistant professor in networking at the Department of Computer Science, University of California, Davis, from September 1998 to June 1999. He was a William and Leila Fellow at Stanford University in 1993–94. At Princeton, he was the recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence in 1993. He is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa.