International Journal of Innovative Computing, Information and Control
Volume **8**, Number **7(A)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 4719–4748

SEMPRE: SECURE MULTICAST ARCHITECTURE USING PROXY RE-ENCRYPTION

Yun-Peng Chiu[1], Chun-Ying Huang[2] and Chin-Laung Lei[1]

[1]Department of Electrical Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan
frank@fractal.ee.ntu.edu.tw; lei@cc.ee.ntu.edu.tw

[2]Department of Computer Science and Engineering
National Taiwan Ocean University
No. 2, Peining Rd., Keelung 20224, Taiwan
chuang@ntou.edu.tw

Received February 2011; revised June 2011

Abstract. *The goal of a secure multicast communication environment is to ensure that only valid members belonging to the multicast group can decrypt data. A simple solution adopted by many previous studies is to use a "group key" that is shared by all group members. The sender uses the group key to encrypt the multicast data, and the receivers decrypt the data with the same key. However, the procedure may incur the so-called "1 affects n problem", whereby the action of one member affects the whole group. This is the source of scalability problems. Moreover, from an administrative perspective, it is desirable to confine the impact of membership changes to a local area. In this paper, we propose a novel secure multicast architecture that reduces the impact of the 1 affects n problem significantly by exploiting a cryptographic primitive, "proxy re-encryption". Therefore, we call the proposed secure multicast architecture Sempre (SEcure Multicast architecture using Proxy Re-Encryption). Proxy re-encryption allows intermediate routers to convert the ciphertext encrypted with one key to ciphertext encrypted with another key, without revealing the private key or the plaintext. If the intermediate routers are given proper keys, they can provide separation between subgroups and thereby achieve the goals of scalability and containment. Successful containment reduces the 1 affects n problem significantly. We also compare several related schemes, and discuss some security problems that we identified in them. Existing schemes that use similar techniques only use asymmetric-key algorithms, but the computational costs of the algorithms mean that the schemes are infeasible in practice. Our scheme combines asymmetric-key and symmetric-key algorithms, so it is practical for real-world applications.*

Keywords: Secure multicast, Multicast key management, Proxy re-encryption

1. Introduction.

Since the emergence of multicast communications in the late 1980s [1, 2], the issue of secure multicast communications has been addressed frequently in the literature. Rafaeli and Hutchison provide a detailed survey of works on secure multicast [3].

Several approaches utilize a group key that is shared by all group members. The sender uses the group key to encrypt the multicast data, and all valid members use the same group key to decrypt the data. However, the group-wise key may incur the so-called "1 affects n problem" [4], which means that the action of one member affects the whole group. More specifically, because the group key is known by all members, whenever a member joins or leaves the group, the remaining members of the group must acquire a new group key.

To build a practical and secure multicast architecture, we focus on scalability and containment issues. Scalability means that the processing overhead of each security operation should be minimized in terms of the number of group members. Containment means that a security breach that occurs in one subgroup does not affect other subgroups. We adopt two techniques to address these issues. The first distributes the computational load to intermediate routers such that the whole architecture is scalable. The second technique provides the keying material based on the topology of the multicast network. The dependency ensures that security breaches can be contained.

A naive way to achieve containment is to allow an intermediate router to decrypt traffic (from an upstream router) with one key and then re-encrypt the same traffic with another key [4]. This method requires full trust in the routers because they have the ability to decrypt plain text, which is an undesirable feature.

In this paper, we propose a novel secure multicast architecture for large and dynamic groups. Specifically, we focus on the one-to-many communication pattern and exploit a cryptographic primitive called "proxy re-encryption". By using the primitive, a proxy can convert the ciphertext for one person into the ciphertext for another person without revealing the secret decryption keys or the plaintext. Therefore, we call the proposed secure multicast architecture Sempre (SEcure Multicast architecture using Proxy Re-Encryption). Our multicast model also considers routers; hence, it is more realistic than schemes based on logical trees, e.g., LKH-based schemes [5, 6]. In our architecture, routers play the role of proxies in proxy re-encryption. Because different parts of the network have different ciphertext and encryption keys, the impact of a security breach can be limited to a local area. Therefore, the goals of scalability and containment can be achieved.

In multicast security, forward secrecy and backward secrecy are usually considered. Here, we follow the definitions in [7]. Forward secrecy means that a passive adversary who knows a subset of old group keys cannot infer subsequent group keys. On the other hand, backward secrecy means that a passive adversary who knows a subset of new group keys cannot infer previous group keys. Thus, to achieve forward and backward secrecy, rekeying is necessary when a member joins or leaves the group.

Mukherjee and Atwood proposed a multicast key management scheme called SIM-KM, which also exploits proxy re-encryption [8, 9, 10]. However, we found that the scheme fails to achieve forward secrecy. Further details are provided in Section 3.3. In contrast, our architecture can achieve forward secrecy.

The remainder of this paper is organized as follows. We introduce basic concepts and provide a historical perspective of proxy re-encryption in Section 2. Related works are reviewed in Section 3. Section 4 describes the proposed secure multicast architecture based on proxy re-encryption. The properties of proxy re-encryption systems related to our architecture are discussed in Section 5. We also compare the proposed scheme with similar approaches. Section 6 contains some concluding remarks.

2. **Proxy Re-Encryption.**

Proxy re-encryption is the core technique in our architecture. In Section 2.1, we describe its basic concepts and review some proxy re-encryption schemes. Section 2.2 discusses symmetric-key based proxy re-encryption schemes.

2.1. **Basic concepts and historical review.**

The notion of proxy re-encryption was first introduced by Blaze, Bleumer and Strauss in 1998 [11]. The basic idea is that a proxy, given a proxy key, could convert the ciphertext for one person into the ciphertext for another person without revealing the secret decryption keys or the plaintext. Figure 1 shows the concept of proxy re-encryption. In this figure, S is the sender, P is the proxy, and A and B are two users.

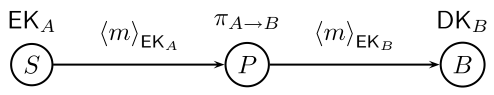

EK_A
$\langle m\rangle_{\mathsf{EK}_A}$
$\pi_{A\to B}$
$\langle m\rangle_{\mathsf{EK}_B}$
DK_B
S
P
B

FIGURE 1. Proxy re-encryption

In traditional asymmetric-key encryption schemes, a message m encrypted using A's public key EK_A could only be decrypted using A's private key DK_A. In contrast, in a proxy re-encryption scheme, a new role, proxy P, is introduced. P is given a proxy key $\pi_{A\to B}$, and P could convert the ciphertext originally for user A to a message which could be decrypted using user B's private key, DK_B. In Figure 1, the notation $\langle m\rangle_k$ represents a message m is encrypted by a key k, using an asymmetric-key algorithm. Here A delegates the right to decrypt the ciphertext for him/her to B; therefore A is called the delegator, and B is called the delegatee. It is infeasible for P to decrypt the ciphertext, and to gain information about A's and B's private keys.

Assume $\mathsf{Enc}(m, e)$ denotes the asymmetric-key encryption algorithm for message m and encryption key e; $\mathsf{Dec}(c, d)$ denotes the decryption algorithm for ciphertext c and decryption key d; $\mathsf{Pre}(c, \pi)$ is the proxy re-encryption algorithm for ciphertext c and proxy key π. The correctness of a proxy re-encryption scheme could be expressed as:

$$\mathsf{Dec}(\mathsf{Pre}(\mathsf{Enc}(m, \mathsf{EK}_A), \pi_{A\to B}), \mathsf{DK}_B) = m.$$

Since the notion of proxy re-encryption was introduced, several works have been proposed in this area. Here we briefly review some of them.

In Blaze, Bleumer and Strauss's work [11], they defined the term "atomic proxy cryptography", where "atomic" means that the proxy converts ciphertext without revealing the plaintext. In other words, it is not necessary for the proxy to decrypt and then encrypt. "Atomicity" is the central idea of proxy re-encryption. In addition to be more secure, it is also more efficient than the "decrypt and then encrypt" method.

In Ivan and Dodis's work [12], they provided generic solutions for proxy re-encryption and proxy signature. The notions of unidirectional and bidirectional proxy cryptography are discussed.

In [13], Ateniese et al. discussed various security properties of proxy re-encryption schemes. Their scheme is pairing-based.

Green and Ateniese extended the notion of proxy re-encryption to the area of identity-based encryption [14], that is, the identity of the receiver is used as the public key when the sender encrypts messages. They also discussed multi-use proxy re-encryption, that is, ciphertext could be re-encrypted for many times.

2.2. **Symmetric-key based proxy re-encryption.**

Although currently most proxy re-encryption schemes are based on asymmetric-key algorithms, there are also few studies based on symmetric-key algorithms [15, 16]. They provided general ways to construct proxy re-encryption schemes using symmetric-key algorithms. In our architecture, we need multi-use proxy re-encryption schemes, which could re-encrypt ciphertext for many times, so here we only describe how to construct a multi-use proxy re-encryption scheme.

As shown in Figure 2, A, B and P represent a sender, a receiver and a proxy, respectively. Assume each pair of P, A and B shares a key between them. More specifically, assume k_{AB} is the shared key between A and B. Similarly, assume k_{AP} is the shared key between A and P, and k_{PB} is the shared key between B and P. The notation $\{m\}_k$ represents a message m is encrypted by a key k, using a symmetric-key algorithm.

When user A wants to transmit some data to user B with help of proxy P, their operations are described as follows. A first encrypts a message m using k_{AB}, and then

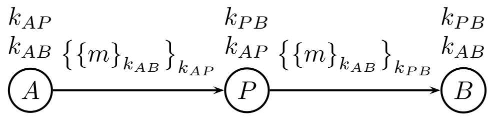


FIGURE 2. Symmetric-key based proxy re-encryption

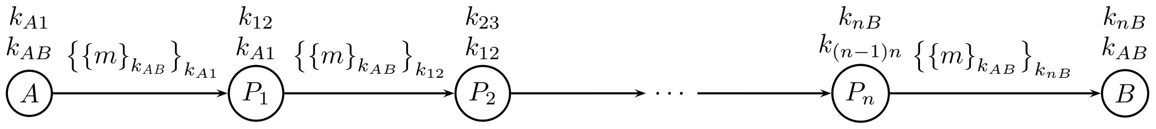


FIGURE 3. Multi-use symmetric-key based proxy re-encryption

encrypts again using k_{AP}. A then transmits this doubly encrypted message to P. After receiving the doubly encrypted message, P first decrypts it using k_{AP}, and then encrypts it again using k_{PB}. Then P sends this to B. B first decrypts the message using k_{PB}, and then decrypts it again using k_{AB}. Since P does not have k_{AB}, it is infeasible for P to decrypt data transmitted from A to B.

We easily extended the above notions to multi-use. Assume now we have n proxies, P_1, P_2, $\ldots$, P_n, between A and B, as shown in Figure 3. Keys owned by some entity are shown above that entity. For example, A have k_{A1}, and P_1 also have the same key. This means k_{A1} is shared between A and P_1. Like the previous single proxy case, A and B share k_{AB}. A does the same with the previous case. P_1 decrypts the message from A using k_{A1}, and then encrypts it again using k_{12}, which is the shared key between P_1 and P_2. Similarly, P_2 decrypts the message from P_1 using k_{12}, and then encrypts it again using k_{23}, which is the shared key between P_2 and P_3. The same operations are repeated until P_n, Finally, B first decrypts the message using, k_{Bn}, the shared key between B and P_n, and then decrypts it again using k_{AB}. Thus, a multi-use symmetric-key based proxy re-encryption scheme could be implemented.

3. Related Works.

Many studies about secure multicast have been published over these years. In this section, we review some existing studies in this problem area. Logical key hierarchy (LKH) may be the most representative research in this area; many studies followed their methodology and tried to enhance it. The cipher sequences (CS) framework tries to solve the multicast security problem using a different methodology. The most important advantage of CS is containment. Containment allows easy management and scalability. We also discuss SIM-KM, which also makes use of proxy re-encryption. In the last part of this section, we briefly describe some other schemes with different approaches.

3.1. Logical key hierarchy.

Logical key hierarchy (LKH) was separately proposed by Wallner et al. [5] and Wong et al. [6]. In this approach, all group members form a *logical* tree. The root node represents the group key shared by all group members, the leaf nodes are members, and each inner node represents a key encryption key (KEK). Besides the group key, a member also has a set of KEKs, including the KEK of each node in the path from its parent to the root. For example, in Figure 4, member u_5 has k_5, k_{56}, k_{58}, and the group key, k. Therefore, in a balanced binary tree, a member has $(\log_2 \mathcal{N}) + 1$ keys, where $\mathcal{N}$ is the group size, and $\log_2 \mathcal{N}$ is the height of the tree. When rekeying is needed, these KEKs are used to encrypt new KEKs. For example, if member u_5 leaves the group, the keys it knows should be changed. Therefore, new KEKs k'_5, k'_{56}, k'_{58} and the new group key k' are generated. These new keys are encrypted using KEKs and transmitted

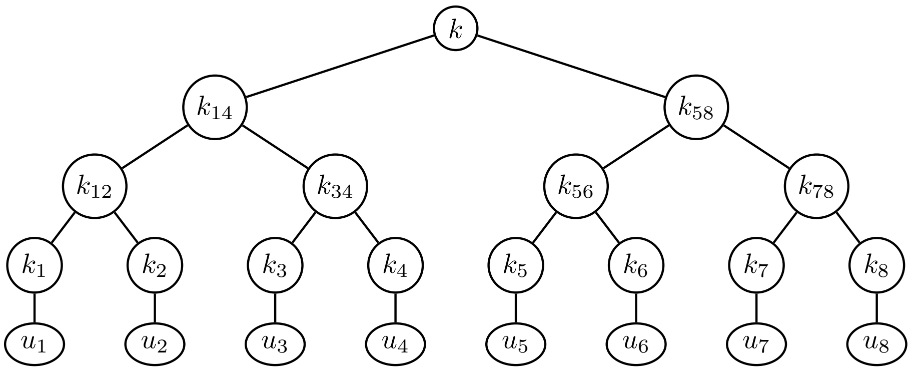


FIGURE 4. An LKH tree

to the remaining members. The key server encrypts new k'_{56} using k_6, and encrypts new k'_{58} using k'_{56} and k_{78}, respectively. Then k' is encrypted using k'_{58} and k_{14}, respectively. Finally, these encrypted keys are multicast to the whole group. All remaining members could get new KEKs and the group key from these encrypted keys.

In LKH-based schemes, they only consider logical trees, and the relationship between routers and members is not considered. Thus, it is difficult to provide containment in LKH-based schemes.

3.2. **Cipher sequences.** The cipher sequences framework (CS) was proposed by Molva and Pannetrat [17]. By distributing secret functions to intermediate nodes, the keying material has a dependency on the multicast network topology. Therefore, containment of security exposures is assured. Intermediate nodes could be routers or application proxies, and this depends on how multicast is implemented. Since in this paper we focus only on network level multicast, an intermediate node of CS is a router.

For example, Figure 5 depicts a simple tree with five cipher sequences. S is the sender, R_i is an intermediate node, and M_i is a leaf node connected to R_i. Assume S_0 is the information to be multicast. Each intermediate node R_i is assigned a secret function f_i. A node R_i receives multicast data from its parent node R_j, computes $S_i = f_i(S_j)$, and forwards the result S_i to its children. A leaf eventually receives S_n^i. Each leaf is given a reversing function h_i, and it uses h_i to get the original multicast data by calculating $S_0 = h_i(S_n^i)$.

We follow one cipher sequence from the root node to leaf M_5. The sender S multicasts message S_0. The router attached to the sender, R_0, computes $f_1(S_0)$ and sends the result to its children inner nodes, R_1 and R_2. R_2 receives $S_1^4 = f_1(S_0)$, computes and sends $f_5(S_1^4)$ to R_5. Then R_5 receives $S_2^4 = f_5(S_1^4)$ and sends $f_6(S_2^4)$ to leaf M_5. Finally, leaf M_4 receives $S_3^4 = f_6(S_2^4)$ and recovers the original multicast data by computing $S_0 = h_4(S_3^4)$.

When a member joins or leaves, the key server generates a new secret function and a corresponding reversing function. The key server sends the new secret function to the last inner node, which connects to the joining/leaving member. And then the new reversing function is sent to all remaining members belonging to the same leaf node.

3.3. **SIM-KM.** Mukherjee and Atwood proposed a multicast key management scheme, SIM-KM, exploiting the proxy re-encryption technique in [8, 9, 10]. In their scheme, there are the group manager (GM), group controllers (GCs) and participants (members). When a group is created, a node is set up as the GM. The GM is configured with group and access control information, and it generates encryption/decryption keys. Moreover, in a multicast tree, there may be several GCs, and each GC is associated with a subtree of

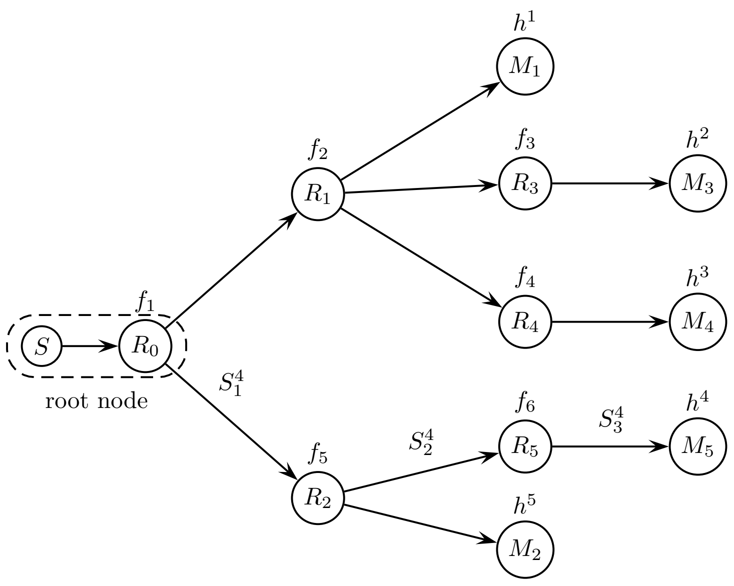


Figure 5. A CS tree

the distribution tree. GCs perform key management functions and transform ciphertext using proxy re-encryption. Participants join the GC nearest to them, and get keys from the GC. Group controllers are classified as CGCs (control group controllers) and DGCs (data group controllers). A CGC is fully trusted. It has access to the group lists, and it could authenticate and grant access to members. It may also transform multicast data. And a DGC, which is untrusted, could only transform multicast data.

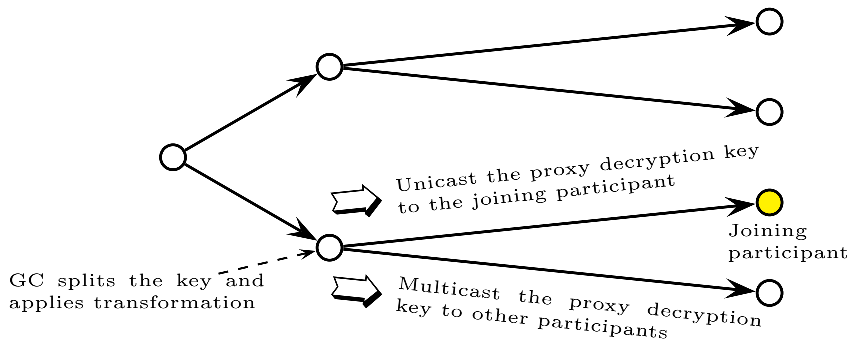
When rekeying is required, the GC "splits" the group decryption key to the "proxy re-encryption key" and the "proxy decryption key". For example, in the unidirectional Elgamal encryption scheme, the group decryption key x is split into x_1 and x_2, such that $x = x_1 + x_2$. Then the GC applies transformations using proxy re-encryption.

When a member joins, the GC sends the proxy decryption key to the joining participant over a secure channel protected by a shared key K_g, and multicast the proxy decryption key to other members. When a member leaves, the GC sends the proxy decryption key to the remaining participants by one of the following two intuitive methods: (a) unicasting the proxy decryption key to each participant over separate secure channels, or, (b) encrypting the proxy decryption key using each participant's K_g and multicasting an aggregated message. Compared with LKH, these methods are intuitive, because the costs of both methods are linear to the number of local participants. Figure 6 shows the basic operations when a participant joins or leaves the group in SIM-KM.

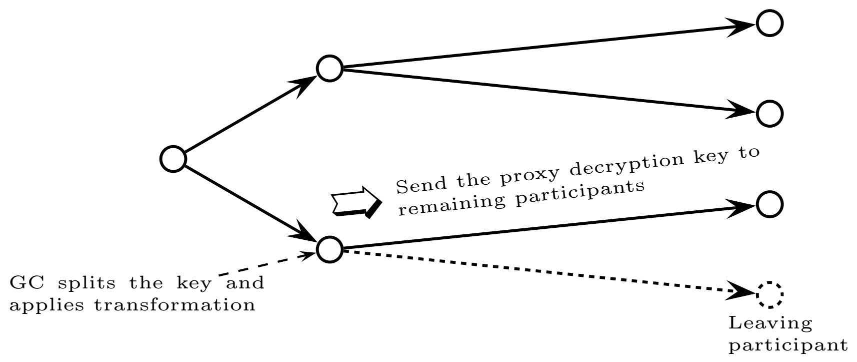
The GM periodically changes the encryption/decryption keys used. During a periodic rekeying event, the GM sends the new encryption key to the sender. Meanwhile, it also sends the new decryption key, which is encrypted by the old encryption key, to all the participants and trusted controllers by a single multicast.

In this scheme, a group key is still used. Proxy re-encryption is used only between a membership change and the next periodic rekeying. After a periodic rekeying event, a new group key is used and transformation is stopped. Please note that when transformation is stopped, SIM-KM uses the Elgamal encryption algorithm to transmit multicast traffic. Moreover, in their scheme, intuitive methods are used to deliver the proxy decryption key to participants and bring a large burden to GCs.

Furthermore, we found some security problems in SIM-KM. When membership changes, the group key is not changed until the next periodic rekeying. If a member is expelled


Unicast the proxy decryption key to the joining participant
Joining participant
GC splits the key and applies transformation
Multicast the proxy decryption key to other participants

(a) A participant joins


Send the proxy decryption key to remaining participants
GC splits the key and applies transformation
Leaving participant

(b) A participant leaves

FIGURE 6. SIM-KM operations

from the group, it still could use the old group key and decrypt multicast traffic, as long as it intercepts encrypted multicast data from another subtree which is not involved with proxy re-encryption. Moreover, according to the above scenario, a more serious problem happens. During a periodic rekeying event, the GM simply sends the new decryption key to all participants by a single multicast. Since the expelled member could decrypt multicast traffic, it could also decrypt the new group key. Therefore, forward secrecy is not achieved.

The author claimed if perfect forward secrecy is required, rekeying is required immediately after a member leaves. However, with help from an eavesdropper, the expelled member could still obtain the new group key. This eavesdropper just receives and stores encrypted multicast data from another part of the multicast tree, and it is not necessarily a member. Once the eavesdropper stores the rekeying message, which contains the new group key encrypted by the old group key, the expelled member could decrypt the new group key from the stored rekeying message.

In order to prevent a member from colluding with a GC, the authors suggested that the sender itself also transforms ciphertext like a GC [10]. However, this remedy does not fix the forward secrecy problem. It just prevents a member from impersonating the sender, but a member still has the right to decrypt.

3.4. **Other related schemes.** In this subsection, we describe some related schemes. These schemes may use different approaches or have different focuses, so we only introduce their basic concepts without delving into details.

Iolus was proposed by Mittra in [4]. Iolus separates the whole multicast group into independent subgroups. In Iolus, a router decrypts encrypted multicast traffic from its upstream router, and then encrypts again for its downstream routers. Therefore, a router could read confidential multicast data, which is an undesirable feature.

Huang et al. proposed a key composition framework in [18], and it is fundamentally different from our scheme. In [18], a group member uses a key composition protocol to obtain the decryption key, which is contributed by the sender and the proxies. Hence, no centralized role is required to maintain multicast tree topology and proxy key assignment. However, it still uses a symmetric key to encrypt messages in the message delivery process. Therefore, during a periodic rekeying event, this key is also required to change. The 1 affects n problem still exists. So it has the same drawback with SIM-KM, that is, no containment is provided.

Hur et al. proposed an Elgamal-based scheme [19]. In their scheme, a proxy (router) shares a proxy key with its downstream proxies and members. And each proxy transforms ciphertext according to shared proxy keys. This actually violates the proxy re-encryption model used in asymmetric-key based proxy re-encryption algorithms. Furthermore, their scheme still uses a group key. When a member joins or leaves, the group key should be changed. Proxy re-encryption is used to transmit key update messages to all members. Thus, the 1 affects n problem still exists. They claimed their main contribution is to eliminate the centralized KMC (key management center), but the fact is that the KMC's job is done by the sender and proxies. Moreover, proxies need to trust each other, because they need communicating the keying material. In our scheme, a proxy just communicates with the KMC for the keying material. Proxies are only responsible for data transmission, not key management. This methodology simplifies not only the protocol design and implementation, but also management. Besides, the task of our KMC could be easily distributed over several KMCs, and the single point of failure problem could be eliminated. We will discuss this later. Some weaknesses of this paper were found in [20]. The main reason of these weakness is that the old group key is encrypted using the new group key. Therefore, Hur et al.'s paper could not provide forward and backward secrecy.

Some studies, such as [21, 22, 23], exploit secret sharing techniques. Briefly speaking, secret sharing is a technique that a secret being divided into several shares. Each participant obtains a share. And a set of shares is required to reconstruct the secret.

4. The Proposed Scheme: Sempre.

In this section, we describe our secure multicast architecture making use of proxy re-encryption. First, the multicast model and the system architecture of our scheme are described in Section 4.1. Then we describe our algorithm for key assignment in Section 4.2 and the rekeying algorithm in Section 4.3.

4.1. Multicast model and system architecture.

In a multicast routing protocol, routers form a multicast tree to transmit multicast traffic. In this paper, we only consider source-based multicast trees [24, 25], that is, the routing protocol builds a spanning tree for each source (sender). And in each tree, the router connected to the sender is the root. Each spanning tree could be treated independently, because each tree works independently. Without loss of generality, we only consider one multicast tree in our protocol.

A multicast network could be represented as a graph of routers. Figure 7 depicts an example of a multicast network. As shown in Figure 7, there are three multicast trees. Both Tree 1 and Tree 3 belong to group G_1. In this group, there are two senders, S_1 and S_3. S_1 is the sender of Tree 1, and S_3 is the sender of Tree 3. Group G_2 has only one sender, S_2, and S_2's multicast tree is Tree 2. The notation (G, S) used in Figure 7 is the identity of the pair (group, sender).

In multicast tree topology, a "node" is a router, and an "edge" is a link between two routers. Every router has one upstream edge (from the multicast sender), and has zero

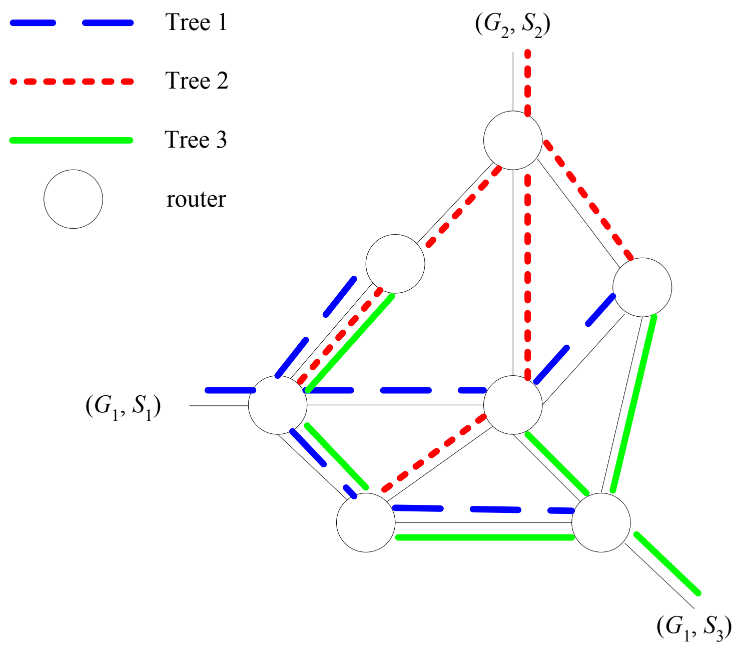


FIGURE 7. Multicast model

or more downstream edges (toward members). Moreover, every router may have group members directly connected in its local area network. These members are called "local members", and the whole of these local members could also be called a "local subgroup". For simplicity, local members are not shown in Figure 7.

Figure 8 shows the block diagram of our architecture. The control plane consists of entities that control security related information, and the data plane consists of entities that transmit multicast data. Separating two planes makes each entity has a clear and simple role. For example, a sender and routers are only responsible for data transmission, not key management. Therefore, this simplifies not only the protocol design and implementation, but also management. Moreover, the key flow shows how keys are distributed or exchanged between entities, and the data flow shows how multicast data are transmitted.

In the original design of IP multicast, a sender does not have information about receivers [1, 2, 26]. This relieves a sender of membership management and makes IP multicast scalable. In our design, a sender only focuses on sending messages, and it does not involve membership management.

Besides a sender, members, and routers in a multicast network, our architecture introduces a key management center (KMC) and local subgroup controllers (LSCs). A trusted KMC is responsible for key management. By "trusted" we assume that the KMC fully follows our protocol and is not compromised. That is, information on the KMC is not stolen. The KMC generates and distributes keys for entities. The KMC is a part of infrastructure. It is a particular entity set up by network administrators, and it is neither a router nor a member node. Moreover, the KMC knows the topology of multicast trees.

For simplicity, we only discuss the KMC as a single entity; however, the task could be easily distributed among multiple entities in our architecture. Multiple KMCs eliminate the single point of failure problem, and they are more practical than a single unit. For example, a multicast tree may cross multiple administrative domains. Each administrative domain has its own KMC, which is responsible for managing the keys in that domain. Assume there are secure channels between KMCs. KMCs are parts of the infrastructure and they are relative minor in number, so it is reasonable to assume that KMCs could

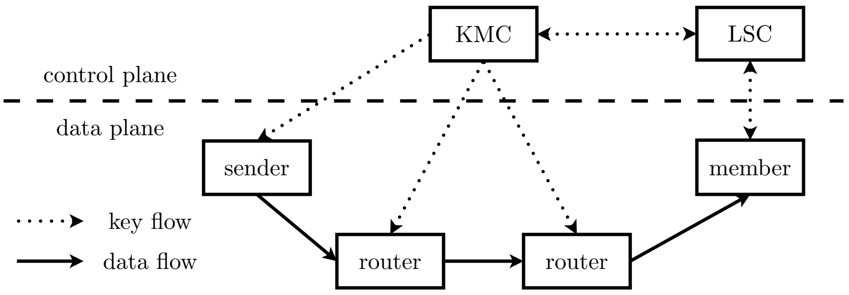


FIGURE 8. System architecture

trust each other. Each KMC just securely exchanges the keying material and the topology information with the KMCs of neighbor domains to generate proper keys.

Assume each local subgroup has its own local subgroup controller (LSC), and it is fully trusted by its local members. An LSC collaborates with the KMC and is responsible for key management in its local subgroup. More details are in Section 4.2.

4.2. **Two modes and key assignment.** Our scheme has two modes: the TEK distribution mode and the data transfer mode. The TEK distribution mode is used to securely distribute TEKs (traffic encryption keys) only, and the data transfer mode is used to transfer multicast traffic, which is encrypted by a TEK. The TEK distribution mode uses asymmetric-key based proxy re-encryption schemes, and the data transfer mode uses symmetric-key based proxy re-encryption schemes. In the following description, Figures 9 and 10 show examples of the two modes respectively. And Figure 11 shows an example of key assignment.

First, we describe the TEK distribution mode. In the TEK distribution mode, a sender transmits an encrypted TEK across its multicast tree. In this mode all encryption and decryption operations are based on asymmetric-key algorithms. The sender first encrypts a TEK using a key given by the KMC, and sends the encrypted key to its local router. Each router in the tree transforms an encrypted TEK according to proxy keys given by the KMC.

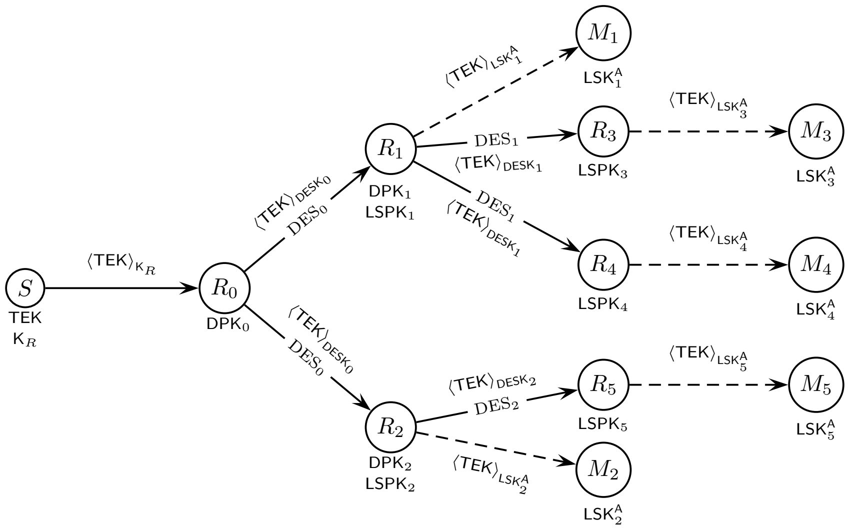
In our scheme, the term "assign a key" means that the KMC generates a key, but only stores this key in its data structure representing the protocol. This key is not really distributed to other entities. On the other hand, if a key should be given to some entity, we explicitly use the term "distribute".

For easy reference, notations of keys used in our scheme are listed in Table 1. In the column "Mode", "TEK" means the key is used in the TEK distribution mode, and "Data" means the key is used in the data transfer mode. Keys used in the TEK distribution mode are asymmetric keys, and keys used in the data transfer mode are symmetric keys. A TEK the only exception, is used to encrypt multicast traffic in the data transfer mode, and it is a symmetric key. Detailed explanations of these keys are given after simple examples of two modes.

Consider a simple multicast tree in Figure 9. S is the sender, R_i is an intermediate node (router), and M_i is the local subgroup connected to R_i. A solid line in the figure represents a router connecting to another router, and a dashed line represents a router connecting to its local members. Here we just describe the notion of the TEK distribution mode, and the details of how to calculate keys for routers are described later. For simplicity, we look at the route from S, along R_0, R_1, and finally to the local subgroups M_1, M_3 and M_4. First, the edge between the sender S and its local router R_0 is assigned a key K_R. And the KMC distributes K_R to S by a secure channel. When the sender S transmits a TEK, which is

TABLE 1. Notations of keys used in our scheme

Key	Meaning	Mode
TEK	Traffic encryption key	TEK, Data
K_R	The root edge's key	TEK
K_0	The key shared by the sender and R_0	Data
DPK	Downstream proxy key	TEK
DESK	Downstream edge set key	TEK
LSPK	Local subgroup proxy key	TEK
PSK	Proxy shared key	Data
LSK^A	Local subgroup key, asymmetric-key based	TEK
LSK^S	Local subgroup key, symmetric-key based	Data



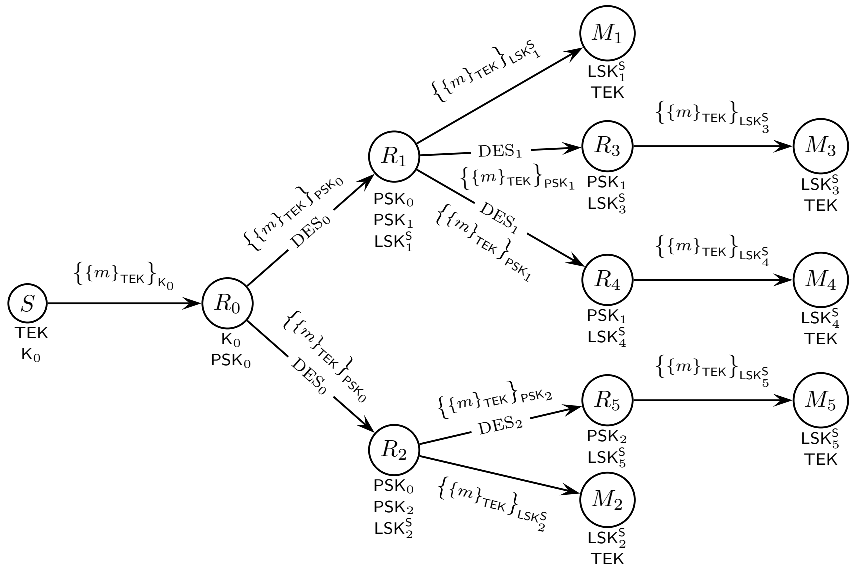
S
TEK
K_R
$\langle\mathsf{TEK}\rangle_{\mathsf{K}_R}$
R_0
DPK_0
$\langle\mathsf{TEK}\rangle_{\mathsf{DESK}_0}$
DES_0
R_1
DPK_1
LSPK_1
$\langle\mathsf{TEK}\rangle_{\mathsf{LSK}^\mathsf{A}_1}$
M_1
$\mathsf{LSK}^\mathsf{A}_1$
DES_1
$\langle\mathsf{TEK}\rangle_{\mathsf{DESK}_1}$
R_3
LSPK_3
$\langle\mathsf{TEK}\rangle_{\mathsf{LSK}^\mathsf{A}_3}$
M_3
$\mathsf{LSK}^\mathsf{A}_3$
R_4
LSPK_4
$\langle\mathsf{TEK}\rangle_{\mathsf{LSK}^\mathsf{A}_4}$
M_4
$\mathsf{LSK}^\mathsf{A}_4$
R_2
DPK_2
LSPK_2
$\langle\mathsf{TEK}\rangle_{\mathsf{DESK}_2}$
DES_2
R_5
LSPK_5
$\langle\mathsf{TEK}\rangle_{\mathsf{LSK}^\mathsf{A}_5}$
M_5
$\mathsf{LSK}^\mathsf{A}_5$
$\langle\mathsf{TEK}\rangle_{\mathsf{LSK}^\mathsf{A}_2}$
M_2
$\mathsf{LSK}^\mathsf{A}_2$

FIGURE 9. An example of the TEK distribution mode

chosen by S itself, S encrypts the TEK using K_R, and then sends to R_0. As we mentioned, routers play the role of proxies in proxy re-encryption in our scheme. R_0 is given DPK_0, which is a proxy key calculated based on proxy re-encryption. Due to proxy re-encryption, the message transmitted on the edge between R_0 and R_1 is effectively encrypted under the key DESK_0, i.e., $\langle\mathsf{TEK}\rangle_{\mathsf{DESK}_0}$. The notation $\langle m\rangle_k$ represents a message m is encrypted by a key k, using an asymmetric-key algorithm. R_1 has local subgroup M_1, so it converts ciphertext for M_1 using LSPK_1, which is also a proxy key calculated based on proxy re-encryption. After proxy re-encryption using LSPK_1, traffic is effectively encrypted under $\mathsf{LSK}^\mathsf{A}_1$. The members in M_1 use $\mathsf{LSK}^\mathsf{A}_1$ to decrypt TEK. R_1 also has downstream routers R_3 and R_4; it converts ciphertext for them using DPK_1 and transmits $\langle\mathsf{TEK}\rangle_{\mathsf{DESK}_1}$ to R_3 and R_4. R_3 and R_4 use their local subgroup proxy keys LSPK_3 and LSPK_4 respectively to convert ciphertext for their local members. The members in M_3 use the local subgroup key $\mathsf{LSK}^\mathsf{A}_3$ to decrypt TEK. And the members in M_4 use $\mathsf{LSK}^\mathsf{A}_4$ to decrypt.

The TEK distribution mode is required in two occasions. One is to periodically rekey the TEK. And the other is when a new member joins the group and its local router has not joined yet. The details of these occasions are further described in Section 4.3.

Now we describe the data transfer mode. In the data transfer mode, a sender transmits encrypted multicast data across its multicast tree. In this mode all encryption and decryption operations are based on symmetric-key algorithms. The sender first encrypts



$\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{K}_0}$
S
TEK
K_0
R_0
K_0
PSK_0
$\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{PSK}_0}$
DES_0
R_1
PSK_0
PSK_1
$\mathsf{LSK}^{\mathsf{S}}_1$
$\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{LSK}^{\mathsf{S}}_1}$
M_1
$\mathsf{LSK}^{\mathsf{S}}_1$
TEK
DES_1
$\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{PSK}_1}$
R_3
PSK_1
$\mathsf{LSK}^{\mathsf{S}}_3$
$\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{LSK}^{\mathsf{S}}_3}$
M_3
$\mathsf{LSK}^{\mathsf{S}}_3$
TEK
R_4
PSK_1
$\mathsf{LSK}^{\mathsf{S}}_4$
$\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{LSK}^{\mathsf{S}}_4}$
M_4
$\mathsf{LSK}^{\mathsf{S}}_4$
TEK
R_2
PSK_0
PSK_2
$\mathsf{LSK}^{\mathsf{S}}_2$
$\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{PSK}_2}$
DES_2
R_5
PSK_2
$\mathsf{LSK}^{\mathsf{S}}_5$
$\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{LSK}^{\mathsf{S}}_5}$
M_5
$\mathsf{LSK}^{\mathsf{S}}_5$
TEK
$\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{LSK}^{\mathsf{S}}_2}$
M_2
$\mathsf{LSK}^{\mathsf{S}}_2$
TEK

FIGURE 10. An example of the data transfer mode

multicast data using a TEK, which is chosen by itself. Routers or the KMC does not know a TEK. And then the sender encrypts the encrypted data again using a key shared between the sender and its local router. Finally the sender sends this doubly encrypted data to its local router. Each router in the tree transforms doubly encrypted data according to the keys given by the KMC. The transformation executed by routers consists of two steps. First a router decrypts using a key shared between it and its upstream router, and then it encrypts again using another key shared between it and its downstream router and/or the local subgroup.

For example, look at Figure 10, which depicts the same multicast tree with Figure 9. We also look at the route from S, along R_0, R_1, and finally to the local subgroups M_1, M_3, and M_4. The sender S first encrypts multicast data m using the TEK, and then encrypts again using K_0, which is a shared key between the sender and R_0. And then S sends this doubly encrypted data, $\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{K}_0}$, to R_0. The notation $\{m\}_k$ represents a message m is encrypted by a key k, using a symmetric-key algorithm. R_0 shares the key PSK_0, which is given by the KMC, with its downstream routers, R_1 and R_2. After receiving doubly encrypted data from S, R_0 decrypts it with K_0, and then encrypts it again with PSK_0. Then R_0 sends $\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{PSK}_0}$ to its downstream routers, R_1 and R_2. Each router first decrypts doubly encrypted data using the PSK shared with the upstream router, and then encrypts again in the following two cases. Case (a), for downstream routers, a router encrypts multicast traffic using the PSK shared with the downstream routers. On the other hand, case (b), for local subgroup members, a router encrypts multicast traffic using its local subgroup key $\mathsf{LSK}^{\mathsf{S}}$. For example, R_1 decrypts the received data using PSK_0. And then R_1 encrypts again using PSK_1 for its downstream routers R_3 and R_4, and encrypts using $\mathsf{LSK}^{\mathsf{S}}_1$ for its local subgroup members, respectively. Finally, a member in a local subgroup first decrypts multicast traffic using the $\mathsf{LSK}^{\mathsf{S}}$ shared with its router, and then decrypts again using the TEK. For instance, members in M_5 first decrypt multicast traffic using $\mathsf{LSK}^{\mathsf{S}}_5$, and then decrypt again using the TEK.

Both the data transfer mode and the TEK distribution mode are required. Only one mode is impractical or unable to work. The data transfer mode is efficient because it is based on symmetric-key algorithms. However, the data transfer mode needs a TEK to

Algorithm 1: Key assignment

```
Input: A multicast tree topology
Output: A key assignment
1  Assign $\mathsf{K}_R$ to the root edge;
2  Assign $R_0$ and the sender a shared symmetric key ($\mathsf{K}_0$);
3  Securely distribute $\mathsf{K}_R$ and $\mathsf{K}_0$ to the sender;
4  Securely distribute $\mathsf{K}_0$ to $R_0$;
5  foreach router R in the tree do
6      if R is marked as PSK-assigned by the upstream router then
7          Assign the PSK to R.
8      end
9      if R has downstream router(s) then
10         Assign the DESK to the downstream edge set (DES);
11         Assign the DPK to R according to the upstream edge key and the DESK;
12         Assign the PSK to R and mark downstream router(s) as PSK-assigned with the
           same PSK;
13     end
14     if R has local subgroup member(s) then
15         Securely obtain the local subgroup keys ($\mathsf{LSK}^\mathsf{A}$ and $\mathsf{LSK}^\mathsf{S}$) from the LSC;
16         Assign the LSPK to R according to the upstream edge key and $\mathsf{LSK}^\mathsf{A}$;
17     end
18     Securely distribute the assigned proxy keys (DPK and/or LSPK) to R;
19     Securely distribute the assigned shared symmetric keys (PSK and/or $\mathsf{LSK}^\mathsf{S}$) to R;
20 end
```

work. It is impossible to use the data transfer mode to distribute a TEK, because it needs another TEK to transfer this TEK. It becomes the chicken or the egg problem of TEKs.

The TEK distribution mode is the answer to the above problem. Because of the nature of asymmetric-key based proxy re-encryption schemes, no prior TEK is required. The previous version of our scheme, [27], only has the TEK distribution mode. However, it is impractical to use asymmetric-key based proxy re-encryption schemes to transfer a large amount of data because of the high costs of asymmetric-key based schemes. Therefore, the combination of the two modes is necessary.

The data transfer mode is similar to Iolus [4]. But in Iolus, routers can obtain plaintext of multicast traffic. In our scheme, the combination of the two modes solves this problem. It is infeasible for routers to decrypt multicast traffic. More details of security analysis are described in Section 5.2.

Because of high costs of asymmetric-key based schemes, CS suggested using their scheme for key distribution, and not for multicast traffic itself. However, if CS is used to transfer a group key, the 1 affects n problem spoils the benefits of CS. The two modes of our scheme solve these problems practically.

After seeing simple examples of two modes, now we formally describe our key assignment algorithm. In our scheme, each multicast tree is assigned a set of keys. The key assignment algorithm is executed by the KMC, and it tells the KMC how to distribute/assign keys for entities. The pseudo code of the key assignment algorithm is given in Algorithm 1.

In our design, we let each local subgroup use its own secure key management protocol. Every subgroup could choose its own key management protocol independent from other subgroups. Since the size of a local subgroup is much smaller than the size of the whole group, the impact of group size to key management protocols is alleviated. Therefore, using LKH schemes, for example, will be acceptable in such circumstances.

Because each subgroup has its own key management protocol, each subgroup should have its own local subgroup controller (LSC). An LSC, who collaborates with the KMC, is responsible for key management in its local subgroup. The same as LKH, the key management center in our scheme is fully trusted, because it knows all the keys. Similarly, an LSC in our scheme is fully trusted by its local subgroup members. This is reasonable because the trust is only limited in a local subgroup.

The KMC assigns an "edge key" for every edge in the multicast tree. These edges keys are only stored in the KMC's data structure, and are not distributed to other entities. The KMC also assigns and distributes keys for each router in the multicast tree. The key assignment algorithm begins with the root edge and the root router. The root edge is the edge connected to the sender. The root router is the root of a multicast tree, and it is connected to the sender via the root edge. First the KMC assigns K_R to the root edge, and assigns the root router and the sender a shared symmetric key (K_0). Then the KMC securely distributes K_0 and K_R to the sender, and securely distributes K_0 to the root router. Only the edge key of the root edge is distributed to the sender, and other edge keys are actually stored in the KMC only.

Because the KMC only processes one router in one iteration of the loop (line 5 in Algorithm 1), the status of downstream routers could not be handled in the iteration of the current router. Therefore, we only mark them for later processing in later iterations. The status "PSK-assigned" means a router is assigned a PSK (proxy shared key) by its upstream router. When an upstream router marks its downstream routers as PSK-assigned, it only tells the KMC that the downstream routers will have a PSK. The assignment of the PSK to downstream routers is actually done when the KMC processes the downstream routers.

If a router R has downstream routers, all downstream edges of this router (the "downstream edge set", DES), are assigned the same downstream edge set key (DESK). Please note the DESK is not distributed; it is stored on the KMC. Then the KMC calculates the DPK (downstream proxy key) according to the upstream edge key and the DESK, and assigns the DPK to R. A router uses the DPK to convert the encrypted TEK for its downstream routers in the TEK distribution mode. Because of proxy re-encryption, the messages transmitted on these downstream edges are effectively encrypted using the DESK. The KMC also assigns the PSK to R. A router uses the PSK to encrypted multicast traffic for downstream routers in the data transfer mode. At the same time, the KMC also marks the downstream routers of R as PSK-assigned for later processing.

Each local subgroup has its own local subgroup keys. Members in the same local subgroup use these keys to decrypt the TEK and multicast traffic from the attached router.

If a router R has local subgroup members, the KMC calculates the LSPK (local subgroup proxy key) according to the upstream edge key and the asymmetric-key based local subgroup key (LSK^A), and assigns the LSPK to R. A router uses the LSPK to convert the TEK for its local members. And local subgroup members use the LSK^A to decrypt the TEK using proxy re-encryption in the TEK distribution mode. Furthermore, a symmetric-key based local subgroup key (LSK^S) is used for a router and its local subgroup members to encrypt/decrypt encrypted multicast traffic in the data transfer mode. The LSC securely distributes the LSK^S to the KMC, and then the KMC securely distributes the LSK^S to the router. These local subgroup keys (LSK^A/LSK^S) are managed by the LSC.

Finally, the assigned proxy keys (DPK and/or LSPK) and the share symmetric keys (PSK and/or LSK^S) are securely distributed to the router.

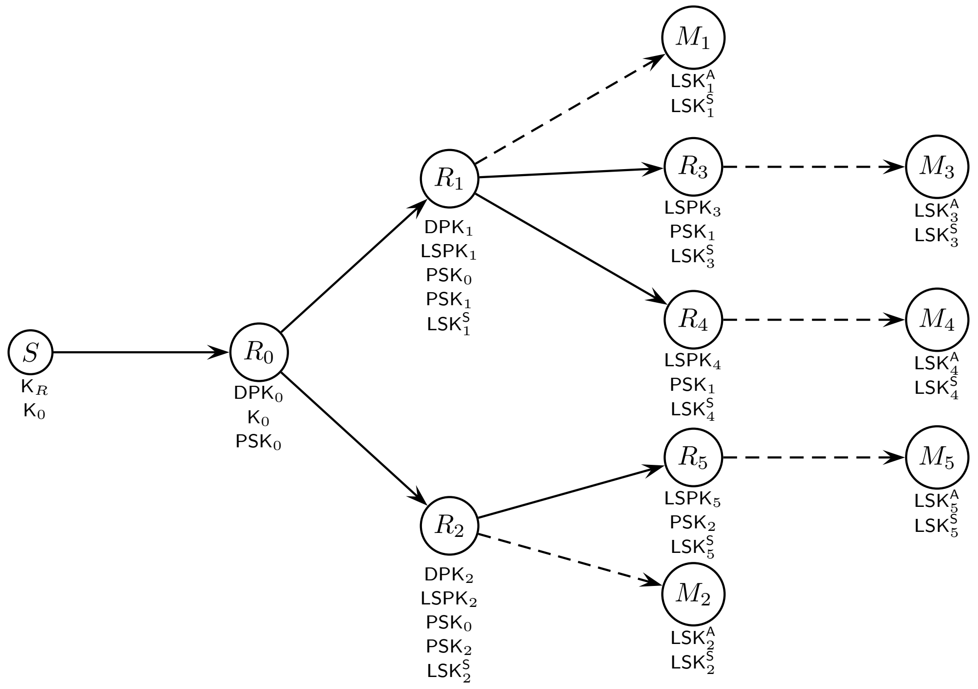
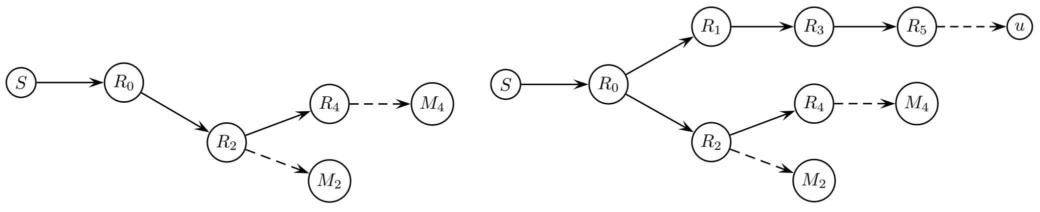

S
K_R
K_0
R_0
DPK_0
K_0
PSK_0
R_1
DPK_1
LSPK_1
PSK_0
PSK_1
$\mathsf{LSK}^{\mathsf{S}}_1$
R_2
DPK_2
LSPK_2
PSK_0
PSK_2
$\mathsf{LSK}^{\mathsf{S}}_2$
M_1
$\mathsf{LSK}^{\mathsf{A}}_1$
$\mathsf{LSK}^{\mathsf{S}}_1$
R_3
LSPK_3
PSK_1
$\mathsf{LSK}^{\mathsf{S}}_3$
R_4
LSPK_4
PSK_1
$\mathsf{LSK}^{\mathsf{S}}_4$
R_5
LSPK_5
PSK_2
$\mathsf{LSK}^{\mathsf{S}}_5$
M_2
$\mathsf{LSK}^{\mathsf{A}}_2$
$\mathsf{LSK}^{\mathsf{S}}_2$
M_3
$\mathsf{LSK}^{\mathsf{A}}_3$
$\mathsf{LSK}^{\mathsf{S}}_3$
M_4
$\mathsf{LSK}^{\mathsf{A}}_4$
$\mathsf{LSK}^{\mathsf{S}}_4$
M_5
$\mathsf{LSK}^{\mathsf{A}}_5$
$\mathsf{LSK}^{\mathsf{S}}_5$

FIGURE 11. Key assignment example

Now we see Figure 11 as an example of how keys are assigned to entities by the KMC. Figure 11 depicts the same multicast tree with Figure 9 and Figure 10. As mentioned, the KMC securely distributes K_0 and K_R to the sender, and securely distributes K_0 to the root router, R_0. R_0 has two downstream edges, one is connected to R_1, and the other is connected to R_2. These edges are the DES of R_0; therefore they are assigned DESK_0. Then the KMC calculates and distributes DPK_0 to R_0, and the KMC also distributes PSK_0 to R_0. R_1 is already assigned a PSK which is the same with R_0; therefore it is given PSK_0. R_1 also has two downstream edges, one is connected to R_3, and the other is connected to R_4. These edges are assigned DESK_1. And R_1 itself is given DPK_1 and PSK_1. R_1 also has local members; thus it is given LSPK_1 and $\mathsf{LSK}^{\mathsf{S}}_1$. R_3 and R_4 are R_1's downstream routers, so they are given PSK_1. Moreover, R_3 and R_4 only have local members. Therefore, R_3 and R_4 are given their respective LSPK and $\mathsf{LSK}^{\mathsf{S}}$. Other routers and local members have similar situations; thus we will not go further here.

4.3. **Rekeying.** Now we describe the rekeying procedures. To provide forward and backward secrecy, a rekeying procedure is required when a member joins or leaves the group. First we discuss general concepts about rekeying. Then we discuss the procedures when a member joins or leaves in Section 4.3.1 and Section 4.3.2, respectively. These procedures maintain the architecture constructed by the key assignment algorithm when membership changes. Furthermore, another type of rekeying, periodic rekeying, is discussed in Section 4.3.3.

When membership changes, the KMC must change and distribute keys for entities in the multicast tree. The local subgroup keys ($\mathsf{LSK}^{\mathsf{A}}$ and $\mathsf{LSK}^{\mathsf{S}}$) should be changed when a member joins or leaves the group. At the same time, the router responsible for this subgroup should also change new proxy keys according to the new local subgroup keys in order to convert ciphertext for local members. Related edge keys are also changed. And the KMC calculates proxy keys according to edge keys.



(a) Without user u (b) With user u

FIGURE 12. A special case where some routers have only one downstream router/member

Moreover, in some cases, a single joining or leaving event may cause one or more routers also to join or to leave the group. As we mentioned, routers form a multicast tree to transmit multicast traffic. When there are downstream routers or local members interested in joining the group, a multicast router should connect itself and its downstream routers to the multicast tree of this group. More specifically, when a member joins the group, if its local router is not in the group yet, the router itself must further join the group. For example, a multicast tree is shown in Figure 12(a), and a user u, who is distant from the current multicast tree, wants to join the group. First u joins the group by connecting to its local router R_5. R_5 is not in the multicast tree yet; therefore, it connects to its upstream router R_3. And then R_3 connects to its upstream router R_1. The final result is shown in Figure 12(b). As we can see, a single joining event may cause a chain reaction, that is, one or more routers also join the group. This case could happen especially when a group is newly established.

On the other hand, when a member leaves the group, if its local router has no other local members or downstream routers, this router should also leave the group. Sometimes a router only has one downstream router, as shown in Figure 12(b). In this case, when the only user u leaves R_5, R_5 should also leave the group. Meanwhile, R_3 has no other local members or downstream routers, so it should be also removed from the multicast tree. Finally R_1 also leaves the group. Then the multicast tree is shown in Figure 12(a). This is a chain reaction (one or more routers leave) when a single member leaves the group.

A single joining or leaving event may cause one or more routers also to join or to leave. In these circumstances, joining or leaving routers always form a chain. That is, some upstream router has only one downstream router, which also has only one downstream router, and so on. As shown in Figure 12(b), R_1, R_3 and R_5 form a router chain. And the router directly connected to the joining/leaving member is always the lowest-level one in the chain. The whole chain of routers is handled together in our algorithm. When a router joins or leaves the group, related edge keys must be changed; routers related with these edge keys must also change new proxy keys. First the highest-level router (the router nearest to the sender in the chain) should be found. Since the KMC knows the topology of multicast trees, it is trivial for it to find the highest-level router in the chain. Then our algorithm processes these routers from the highest-level router.

Please note transformations of multicast trees are maintained by underlying routing protocols. Our architecture just responds to changes of an underlying multicast tree, and it does not change a multicast tree.

4.3.1. *Member joining.* The procedures of member joining are shown in **Procedure** `Member-Join` and **Procedure** `Router-Join`. If the local router has already joined the group when a member joins, only **Procedure** `Member-Join` is executed. Otherwise, if a member joining causes a chain reaction, that is, one or more routers also join the group,

```
Procedure Member-Join
  Input: An old key assignment and the identifier of the router where membership changes
  Output: A new key assignment
1 The joining member sends a joining request to its local router;
2 if the local router has not joined the group then
3     The router sends a joining request to the group;
4     The KMC executes Procedure Router-Join according to the resultant multicast tree;
5     The KMC notifies the sender to initiate the TEK distribution mode;
6 end
7 The LSC generates a new $\mathsf{LSK}^{\mathsf{A}}$ and a new $\mathsf{LSK}^{\mathsf{S}}$;
8 The LSC securely distributes the $\mathsf{TEK}$, the new $\mathsf{LSK}^{\mathsf{A}}$, and the new $\mathsf{LSK}^{\mathsf{S}}$ to the joining
  member using its local key management protocol;
9 The LSC securely distributes the new $\mathsf{LSK}^{\mathsf{A}}$ and the new $\mathsf{LSK}^{\mathsf{S}}$ to the existing members
  using its local key management protocol;
10 The LSC securely distributes the new $\mathsf{LSK}^{\mathsf{A}}$ and the new $\mathsf{LSK}^{\mathsf{S}}$ to the KMC;
11 The KMC calculates the $\mathsf{LSPK}$ and securely distributes the $\mathsf{LSPK}$ and the $\mathsf{LSK}^{\mathsf{S}}$ to the
  local router;
```

Procedure `Member-Join` would invoke **Procedure** `Router-Join`. These procedures are executed by the KMC to change the key assignment of the new multicast tree. In this case, the KMC handles these routers one by one from the highest-level router to the lowest-level router.

In **Procedure** `Member-Join`, the LSC generates new keys and securely distributes them to the joining member and the existing local subgroup members using its local key management protocol. If the local router has joined the group, the LSC already has a copy of the TEK. In contrast, if the local router has not joined the group, the LSC obtains the new TEK from the execution of the TEK distribution mode. Therefore, the joining member u obtains the TEK from the LSC. The LSC also distributes keys to the KMC for calculation of the LSPK. And in **Procedure** `Router-Join`, first the highest-level router of the chain (R_h) and its upstream router (R_u) are found. Keys related to R_h are changed. If R_u has other downstream edges, the edge key, DESK should be changed. And DPK should be changed accordingly. PSK shared by R_u and its downstream routers are changed. On the other hand, if R_u does not have other downstream edges, the KMC just generates DESK, DPK and PSK for R_u and R_h. Then routers along the chain are processed sequentially. The KMC generates DESK, DPK and PSK for each router in the chain.

Now we look at an example where a chain of routers joins, as shown in Figure 13. It is just like the case from Figure 12(a) to Figure 12(b). In this example, user u joins, and a chain of routers, including R_1, R_3 and R_5, also join the group. Here R_1 is R_h, and R_0 is R_u. Because R_0 has another downstream edge connected to R_2, it is necessary to change R_0's DESK_0 and DPK_0. So the KMC changes R_2's DPK_2 and LSPK_2. The KMC also changes PSK_0 shared by R_0 and its downstream routers, and securely distributes the new PSK_0 to R_0, R_1, and R_2. Then the KMC generates new keys for routers in the chain. First, the KMC assigns DESK_1 to R_1. Then the KMC calculates and securely distributes DPK_1 to R_1. The KMC also generates and securely distributes PSK_1 for R_1 and its downstream router, R_3. Similarly, the KMC securely distributes DPK_3 and PSK_3 to R_3. Finally, R_5 is the last router in the chain, that is, it directly connects to the joining member u. The LSC of u generates a new $\mathsf{LSK}^{\mathsf{A}}$ and a new $\mathsf{LSK}^{\mathsf{S}}$. The KMC securely distributes LSPK_5, $\mathsf{LSK}_5^{\mathsf{S}}$ and PSK_3 to R_5. The joining member u is given local subgroup keys $\mathsf{LSK}_5^{\mathsf{A}}$ and $\mathsf{LSK}_5^{\mathsf{S}}$, which are generated by the LSC. u is also given the TEK from the LSC. In Figure 13, keys with a star ($\star$) and shown in boldface are keys required to be

```
Procedure Router-Join
  Input: An old key assignment, a multicast tree topology, and a list ($\mathcal{L}$) of identifiers of
         joining routers caused by a single membership change
  Output: A new key assignment
1  if the first router joins the group then // extreme case, group created
2      Create key management related information for this group;
3  end
4  Find the highest-level router $R_h$ in $\mathcal{L}$;
5  Assume $R_h$'s upstream router is $R_u$;
6  if $R_u$ has other downstream edges then
7      Change $R_u$'s DESK;
8      Change $R_u$'s DPK and securely distribute it to $R_u$;
9      Change the PSK shared by $R_u$ and its downstream routers, and securely distribute it to
       $R_u$;
10     Mark downstream router(s) as PSK-assigned with the same PSK;
11     foreach downstream router $R_d$ connected to $R_u$'s DES do
12         Change $R_d$'s DPK and/or LSPK and securely distribute it to $R_d$;
13         Change $R_d$'s PSK as previously assigned by $R_u$ and securely distribute it to $R_d$;
14     end
15 else
16     Generate a new DESK and calculate the DPK according to the upstream edge key and
       the DESK;
17     Securely distribute the DPK to $R_u$;
18     Generate a new PSK;
19     Securely distribute the PSK to $R_u$ and $R_h$;
20 end
   // begin to process a chain of routers
21 $R = R_h$;
22 while $|\mathcal{L}| > 1$ do
23     Assign $R$'s DESK;
24     Calculate $R$'s DPK and securely distribute the DPK to $R$;
25     Generate a new PSK for $R$ and its downstream router;
26     Securely distribute the PSK to $R$ and mark $R$'s downstream routers as PSK-assigned
       with the same PSK;
27     Remove $R$ from $\mathcal{L}$;
28     $R \leftarrow R$'s downstream router;
29 end
```

changed. Please note that R_4, M_4 and M_2 are not affected. That is, they do not have to change their keys. Here we can see the benefits of containment, and members in other subgroups do not have to change anything. Only members in the same subgroup, some upstream and/or neighbor routers are affected. Thus, the 1 affects n problem is reduced to a much smaller extent.

4.3.2. *Member leaving.* The procedures of member leaving are shown in **Procedure** `Member-Leave` and **Procedure** `Router-Leave`. If the local router still has other members in the local subgroup when a member leaves, only **Procedure** `Member-Leave` is executed. Otherwise, if a member leaving causes a chain reaction, that is, one or more routers also leave the group, **Procedure** `Member-Leave` would invoke **Procedure** `Router-Leave`. These procedures are executed by the KMC to change the key assignment of the new multicast tree. In **Procedure** `Member-Leave`, like **Procedure** `Member-Join`, the LSC generates new keys and securely distributes them to its remaining local subgroup members. The LSC also distributes keys to the KMC. **Procedure** `Router-Leave` is similar to **Procedure** `Router-Join`, but is simpler. When a chain of routers leaves, the KMC

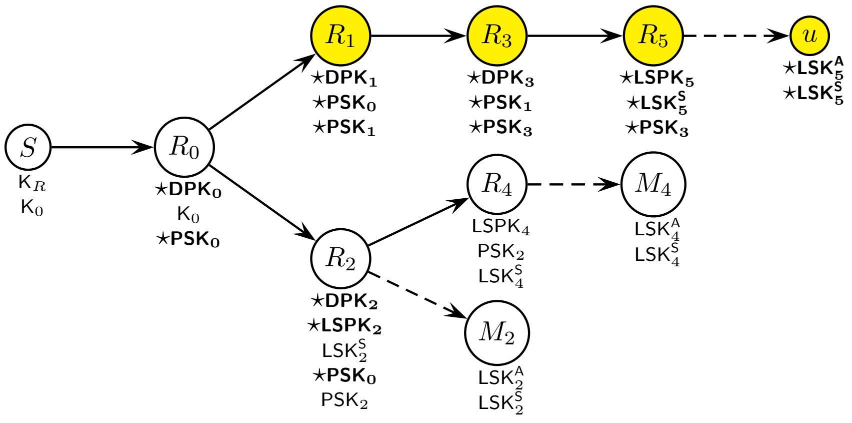


FIGURE 13. A chain of routers joins

Procedure `Member-Leave`

```
   Input: An old key assignment and the identifier of the router where membership changes
   Output: A new key assignment
1  The leaving member sends a leaving request to its local router, or an LSC requests an
   expulsion of a member;
2  if there are other members remaining in the local subgroup then
3      The LSC generates a new $\mathsf{LSK^A}$ and a new $\mathsf{LSK^S}$;
4      The LSC securely distributes the new $\mathsf{LSK^A}$ and the new $\mathsf{LSK^S}$ to the remaining
       members using its local key management protocol;
5      The LSC securely distributes the new $\mathsf{LSK^A}$ and the new $\mathsf{LSK^S}$ to the KMC;
6      The KMC calculates the LSPK and securely distributes the LSPK and the $\mathsf{LSK^S}$ to the
       local router;
7  else
8      The router sends a leaving request to the group;
9      The KMC executes Procedure Router-Leave according to the resultant multicast
       tree;
10 end
```

just processes the highest-level router, just like only the highest-level router leaves. The downstream routers of the highest-level router can be ignored. Keys of lower-level routers are just discarded because they are not used anymore. In the case of Figure 12(b), the KMC processes R_1 as if only R_1 leaves. If R_u does not have other downstream edges,[1] R_u does not need any DESK, DPK, or PSK anymore. Therefore, these keys are just discarded.

Look at Figure 14 for a simple example where one router leaves the group. If all members in M_4 leave the group, R_4 will be deleted from the tree. In this example, only R_4 will leave; thus it is R_h in the algorithm. And R_4's upstream router is R_1, which is R_u in the algorithm. Because R_1 has another downstream edge connected to R_3, first the KMC changes R_1's DESK_1 and DPK_1. The KMC securely distributes DPK_1 to R_1. The old PSK_1 shared by R_1 and R_4 should be changed, and the new PSK_1 is securely distributed to R_1. Since R_1's DESK_1 is changed, R_3's LSPK_3 should be changed too. R_3 gets the new LSPK_3 and the new PSK_1 securely from the KMC. The same with the previous figure, keys with a star ($\star$) and shown in boldface are keys required to be changed. Please note that R_0, the whole R_2 branch, and M_1 are not affected. That is, they do not have to change their keys. Members in other subgroups do not have to change anything. Here we can also see the 1 affects n problem is reduced to a much smaller extent including members in the same subgroup, some upstream and/or neighbor routers.

[1] R_u still has local subgroup members, or it is in the chain, which violates the definition.

```
Procedure Router-Leave
  Input: An old key assignment, a multicast tree topology, and a list ($\mathcal{L}$) of identifiers of
         leaving routers caused by a single membership change
  Output: A new key assignment
1  if the last router leaves the group then // extreme case, group deleted
2      Delete key management related information for this group;
3      return;
4  end
5  Find the highest-level router $R_h$ in $\mathcal{L}$;
6  Assume $R_h$'s upstream router is $R_u$;
7  if $R_u$ has other downstream edges then
8      Change $R_u$'s DESK;
9      Change $R_u$'s DPK and securely distribute it to $R_u$;
10     Change the PSK shared by $R_u$ and its downstream routers, and securely distribute it to
       $R_u$;
11     Mark downstream router(s) as PSK-assigned with the same PSK;
12     foreach downstream router $R_d$ connected to $R_u$'s DES except $R_h$ do
13         Change $R_d$'s DPK and/or LSPK and securely these keys distribute to $R_d$;
14         Change $R_d$'s PSK as previously assigned by $R_u$ and securely distribute it to $R_d$;
15     end
16 else // But $R_u$ still has local subgroup members.
       // Keys for downstream edges are not required anymore.
17     Discard $R_u$'s old keys DESK, DPK, and PSK;
18 end
```

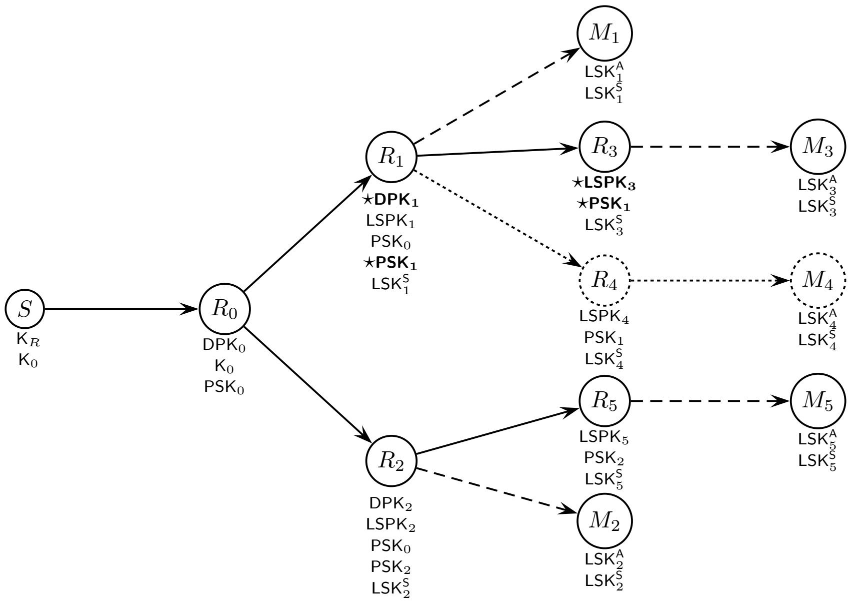


FIGURE 14. A router leaves

4.3.3. *Periodic rekeying.* For better security, the TEK needs to be periodically changed. Here we describe the procedure to change the TEK. The procedure of periodic rekeying is the same with the TEK distribution mode described in Section 4.2. The procedure is initiated by the sender after the sender generates a new TEK. The frequency of periodic rekeying is a tradeoff between security and efficiency. It also depends on the traffic transmission rate [28, 29]. The choice of the periodic rekeying frequency is out of the scope of this paper.

5. **Analysis.**

In this section, first we discuss some properties of proxy re-encryption schemes related to our architecture in Section 5.1. After that, in Section 5.2 we examine the security of our scheme. Then we compare the features and costs of related works and our scheme in Section 5.3 and Section 5.4 respectively.

5.1. **Related proxy re-encryption properties.**

In this subsection, we describe several properties of asymmetric-key based proxy re-encryption systems, and discuss their relationship to our architecture. These properties are originally defined in [13, 14], and they only apply to asymmetric-key based proxy re-encryption systems, not symmetric-key based ones. That is, only schemes used in the TEK distribution mode are related with these properties.

We follow the guidelines defined in RFC 2119, "Key words for use in RFCs to Indicate Requirement Levels" [30] to classify the requirement levels of these properties in our architecture. Three terms of RFC 2119 are used in this paper: "REQUIRED", "MUST NOT" and "RECOMMENDED". If a property is "REQUIRED", that means our architecture requires that property to work. Without that property, the architecture is not secure. "RECOMMENDED" means that property is not a must. But with that property, our architecture has either a higher degree of security if it is a security property, or better performance if it is a performance property. "MUST NOT" means that a property is harmful to security and is absolutely prohibited.

In the following discussions, Alice represents a delegator and Bob represents a delegatee. In our architecture, a delegator means an edge before a router in a multicast tree, and a delegatee is an edge after the same router. In each item of the following properties, first a basic definition is given, and then we discuss the requirement level of that property in our architecture. The following properties which are not specified as performance properties are security properties.

1. Unidirectional: Delegation from Alice to Bob does not allow delegation from Bob to Alice. Obviously, "bidirectional" has the opposite meaning. Specifically, unidirectionality means $\pi_{Bob \to Alice}$ could be derived from $\pi_{Alice \to Bob}$ by a proxy itself. And bidirectionality means $\pi_{Bob \to Alice}$ could be derived from $\pi_{Alice \to Bob}$.

   According to CS [17], there are two categories of cipher sequences: symmetric cipher sequences (SCSs) and asymmetric cipher sequences (ACSs). And an ACS has a higher degree of security than an SCS.

   We find that in the context of proxy re-encryption schemes, a unidirectional proxy re-encryption scheme is an ACS, and a bidirectional proxy re-encryption scheme is an SCS. Therefore, unidirectionality is RECOMMENDED for better security.
2. Original access: Alice can decrypt the re-encrypted ciphertext that was originally sent to her.

   Our architecture MUST NOT use proxy re-encryption schemes with this property. Assume an attacker obtains some edge key. For example, it deciphers the ciphertext on some edge. With original access, the attacker could use this edge key to decrypt re-encrypted ciphertext on downstream edges. On the other hand, without original access, the stolen key could be used to decrypt the ciphertext only on this edge. Therefore, without original access, the damage of a security breach could be confined to the compromised edge.
3. Key optimal: The size of Bob's secret storage remains constant, no matter how much delegation he accepts.

   This is a performance property, and it is RECOMMENDED. In a practical environment, there could be many senders and many groups, so there would be a lot

TABLE 2. Requirement levels of related proxy re-encryption properties

Property	**Requirement level**
Unidirectional	RECOMMENDED
Original access	MUST NOT
Key optimal	RECOMMENDED
Collusion-safe	REQUIRED
Non-transitive	REQUIRED
Multi-use	REQUIRED
Space-optimal	RECOMMENDED

of delegation at the same time. Because all edge keys are stored in the KMC, this property reduces the secure storage required on the KMC.

4. Collusion-safe: It is infeasible for Bob and the proxy to collude to recover Alice's private key.

   This property is REQUIRED. Assume an attacker steals some edge key and compromises the upstream router connected to the edge. Without collusion-safe, the attacker could derive the edge keys of the upstream edge, because he/she already has an edge and the proxy key of the upstream router. Therefore, this property is REQUIRED since it confines the damage of a security breach.
5. Non-transitive: The proxy cannot re-delegate decryption rights by itself.

   This property is REQUIRED in our architecture, because routers are not allowed to re-delegate decryption rights by itself.
6. Multi-use: Ciphertext could be re-encrypted for many times.

   This property is REQUIRED. There are many routers (playing the role of proxies in proxy re-encryption) in the network, and each router transforms ciphertext. Thus it is natural that our architecture needs this property.
7. Space-optimal: No additional communication costs are incurred to support proxy re-encryption.

   This is a performance property. It is natural that less space consumption is preferred, so this property is RECOMMENDED.

Table 2 summarizes the requirement levels of these properties in our architecture.

5.2. **Security analysis.** We analyze the security properties of our architecture in this subsection. By the definition of proxy re-encryption, it is infeasible for a proxy to derive traffic decryption keys with only proxy keys. Therefore, it is infeasible for intermediate routers to decrypt a TEK. Multicast traffic is first encrypted by a TEK, and then is encrypted by a PSK in transmission. None of routers knows a TEK, so it is infeasible for any router to decrypt multicast traffic.

As stated in Section 4.2, each local subgroup uses its own secure multicast key management protocol. We assume each local subgroup key management protocol satisfies secure properties like forward secrecy, backward secrecy and collusion-safe.

We begin our security analysis from the simplest case: a member joins or leaves without changing multicast trees. The local subgroup changes the new local subgroup keys (LSK^A and LSK^S) when a member joins or leaves. Under the assumption of using secure local subgroup key management protocols, when a member joins, it is infeasible for the new member to infer previous local subgroup keys, LSK^As and LSK^Ss. Although the joining member has the current TEK, it still does not have previous LSK^Ss. Thus it is infeasible for the new member to decrypt previous multicast traffic. On the other hand, when a member leaves, the local subgroup keys LSK^A and LSK^S are changed. Only the remaining subgroup members get the new local subgroup keys. Although the left member has the old TEK, it is still infeasible for the left member to decrypt future multicast traffic, $\{\{m\}_{\mathsf{TEK}}\}_{\mathsf{LSK}^{\mathsf{S}'}}$, where $\mathsf{LSK}^{\mathsf{S}'}$ is a new local subgroup key. Because the left member does not have $\mathsf{LSK}^{\mathsf{S}'}$.

Moreover, because the $\mathsf{LSK}^{\mathsf{A}}$ is changed when a member joins or leaves, forward secrecy and backward secrecy of the TEK are achieved. Accordingly, our protocol is secure when a member joins or leaves without changing multicast trees.

Due to proxy re-encryption, multicast traffic and a TEK are encrypted by different keys in different areas. Because each subgroup is independent from each other, subgroup keys in different subgroups are independent. It is infeasible for members in a group to infer subgroup keys in other subgroups. Moreover, when a subgroup changes its subgroup keys, subgroup keys in other subgroups are not affected. Therefore, containment is achieved. Furthermore, a TEK used in our scheme is different from a group key in existing studies. Our TEK does not require changing when membership changes. Therefore, the 1 affects n problem does not occur in our scheme.

Then we discuss a more complex scenario, where a chain of routers joins or leaves caused by a single member joins or leaves. In this case, the PSKs shared by upstream routers and their downstream routers are changed. This prevents the changed routers to transform previous or subsequent multicast data in the data transfer mode. Moreover, related edge keys are also changed. New proxy keys are calculated according to new edge keys and are given to related routers by the KMC securely, so it is infeasible for the new or left routers to infer those keys. Therefore, the new or left routers will not be able to transform previous or subsequent TEKs in the TEK distribution mode. Accordingly, our protocol is also secure when routers change.

In SIM-KM, the new encryption key is encrypted by the old encryption key, forward secrecy is not achieved. But in our scheme, a new encryption key is not related to the old encryption key. Therefore, forward secrecy is preserved.

According to the above analysis, forward and backward secrecy are ensured in our protocol. Moreover, because every local subgroup is isolated with each other, members in the different subgroups gain no more information through collusion.

5.3. **Comparisons of features.** In this subsection, we compare how intermediate nodes are trusted in a scheme and whether a scheme provides containment. The comparisons of features are shown in Table 3. Intermediate nodes mean different entities in different schemes. In LKH, no intermediate nodes are involved, and no containment is provided because of the usage of a group key. In CS and our scheme, an intermediate node is a router. In these schemes, routers are granted limited trust. Because no group key is used in CS, a security breach in one subgroup is confined to that subgroup. Thus, containment is provided in CS. The reasons that Sempre has containment are already discussed in the previous subsection. And in SIM-KM, GCs plays the role of intermediate nodes. For a CGC, full trust is needed, and for a DGC, only limited trust is required. Because SIM-KM uses a group key, when membership changes, the group key known by all remaining members is required to be changed during the next periodic rekeying. The impact of a membership change is just delayed to the next periodic rekeying. Therefore, we still consider that no containment is provided in SIM-KM.

5.4. **Comparisons of costs.** Here we analyze various costs of our scheme, and compare with related works. First, some assumptions are required. We assume LKH is used as the local key management protocol in our architecture. Assume the size of the whole group is $\mathcal{N}$, and the size of a local subgroup is $\mathcal{M}$. Furthermore, $\mathcal{N} \gg \mathcal{M}$. In SIM-KM, the GM is similar to the KMC in our scheme, and a GC is similar to an LSC (local subgroup

Table 3. Comparisons of features

Scheme	**Trust granted to intermediate nodes**	**Containment**
LKH	No intermediate nodes	NO
CS	Limited trust to routers	YES
SIM-KM	CGC: Full trust, DGC: Limited trust	NO
Sempre	Limited trust to routers	YES

Table 4. Notations used in the cost analysis

Notation	**Meaning**
α	Key length of an asymmetric-key algorithm
β	Key length of a symmetric-key algorithm
E_S	Symmetric-key encryption
D_S	Symmetric-key decryption
E_A	Asymmetric-key encryption
D_A	Asymmetric-key decryption
E_R	Asymmetric-key based proxy re-encryption
$\mathcal{N}$	Size of the whole group
$\mathcal{M}$	Size of a local subgroup
$\mathcal{R}$	Number of routers
$\mathcal{S}$	Number of subgroups
$\mathcal{H}$	(CS and Sempre) Length of a chain of routers
$\mathcal{C}$	(SIM-KM) Number of CGs
$\mathcal{T}$	(Sempre) Number of routers who have downstream routers
$\mathcal{P}$	(Sempre) Number of downstream routers connected to a router

controller). So we compare similar entities respectively. Therefore, the costs of routers in SIM-KM are labeled as N/A. Routers and LSCs are not discussed at all in LKH, so their costs in LKH are labeled as N/A.

5.4.1. *Key storage costs.* Now we compare the key storage costs. In the following analysis, α denotes the key length of an asymmetric-key algorithm, and β denotes the key length of a symmetric-key algorithm. Between symmetric-key and asymmetric-key algorithms, LKH only uses symmetric-key ones. The examples of secret and reversing functions given in CS use asymmetric-key algorithms. Moreover, proxy re-encryption schemes used in SIM-KM and the TEK distribution mode of our scheme currently focus on asymmetric-key algorithms. Therefore, all three schemes other than LKH rely on combinations of symmetric-key and asymmetric-key algorithms. Assume secure channels required in each scheme are built using symmetric-key algorithms. More notations are defined in Table 4. And the comparisons of key storage costs are listed in Table 5.

In LKH, the KMC needs to store all logical keys. The number of keys stored depends on the tree structure. For simplicity, here we consider a complete binary tree. The cost is basically of the same order for binary trees. The sender needs two keys: one key for encryption and the other key for the secure channel between it and the KMC. Therefore, the cost of the sender in LKH is 2β. The KMC needs to store $2\mathcal{N}-1$ keys. On the other hand, a member needs to store $(\log_2\mathcal{N}+1)$ keys.

In CS, every member needs a reversing function, and the sender and every router need secret functions. We consider these functions as keys of asymmetric-key algorithms. An LSC has secure channels with each member in its subgroup; thus $\mathcal{M}$ keys are required.

Table 5. Comparisons of key storage costs

Scheme	KMC	Sender	Router	LSC	Member
LKH	$(2\mathcal{N}-1)\beta$	2β	N/A	N/A	$(\log_2 \mathcal{N}+1)\beta$
CS	$(1+\mathcal{R}+\mathcal{S})\alpha+(1+\mathcal{R}+\mathcal{S})\beta$	$\alpha+\beta$	$\alpha+\beta$	$\alpha+(\mathcal{M}+1)\beta$	$\alpha+\beta$
SIM-KM	$\alpha+(1+\mathcal{C})\beta$	$2\alpha+\beta$	N/A	$\alpha+(\mathcal{M}+1)\beta$	$\alpha+\beta$
Sempre	$(1+2\mathcal{S}+2\mathcal{T})\alpha+(2+2\mathcal{S}+\mathcal{R}+\mathcal{T})\beta$	$\alpha+3\beta$	$\leq 2\alpha+4\beta$	$\alpha+(2\mathcal{M})\beta$	$\alpha+(\log_2 \mathcal{M}+2)\beta$

The KMC requires storing all secret and reversing functions. Moreover, it also stores keys for the secure channels with the sender, routers and LSCs, and the result is shown in Table 5.

In SIM-KM, each member stores two keys. One is a group decryption key or a proxy decryption key; the other is the key shared with its GC, K_g. The sender has an encryption key, a proxy key, and a key between it and the KMC, so the cost of the sender is $2\alpha+\beta$. A GC shares K_g with each member, so it stores $\mathcal{M}$ keys. It also needs a key with the GM. During a rekeying event, a GC needs a proxy re-encryption key to apply transformations. Furthermore, the GM needs $\mathcal{C}$ keys to build the secure channels with all subgroup controllers. It also stores the group key and a key shared with the sender.

In our scheme, assume LKH is used as our local subgroup key management protocol, so a member requires $(\log_2 \mathcal{M})$ KEKs. And a member uses KEKs to encrypt the local subgroup keys $\mathsf{LSK}^{\mathsf{A}}$ and $\mathsf{LSK}^{\mathsf{S}}$. Additionally, a member requires a TEK to decrypt multicast traffic. Because of LKH, our LSC stores $2\mathcal{M}-2$ KEKs. An LSC also needs a key for the secure channel with the KMC. Moreover, an LSC stores an $\mathsf{LSK}^{\mathsf{A}}$ and an $\mathsf{LSK}^{\mathsf{S}}$. Every router except the root router has a PSK for its upstream router. If a router has downstream routers, it has a DPK and a PSK. And if a router has local subgroup members, it has an LSPK and an $\mathsf{LSK}^{\mathsf{S}}$. A router also has a key for the secure channel with the KMC. So a router has at most 2 asymmetric keys and 4 symmetric keys. A difference between our model and that of CS is, in CS, members only connect to leaf routers. But in our scheme, members could connect to any router in a multicast tree. Our model is more flexible, and the tradeoff is that a router in our scheme has more keys than CS. The sender has a secure channel with the KMC, a TEK, K_0 and K_R. Therefore, its storage cost is $\alpha+3\beta$. The KMC stores all the keys for the secure channels with the sender, routers, LSCs. It also stores all proxy keys (DPKs and LSPKs), all edge keys (the edge key of the root edge and DESKs), and all $\mathsf{LSK}^{\mathsf{A}}$s and $\mathsf{LSK}^{\mathsf{S}}$s, the result is shown in Table 5.

After analyzing the key storage costs of related schemes, now we compare them from views of different entities. In our scheme, LKH is only used in local subgroups, and $\mathcal{M} \ll \mathcal{N}$. So a member in our scheme stores fewer keys than that in the original LKH schemes. From the sender's view, schemes other than LKH do not differ too much.

For LSCs, our scheme has higher key storage costs than other schemes, and that is the tradeoff of using LKH. As mentioned earlier, a router in our scheme has more keys than a router in CS. That is due to the difference between our model and that of CS.

In the aspect of the KMC, the storage cost of the KMC in LKH is the highest. Our KMC has nearly double key storage cost with CS. And the KMC in SIM-KM has the lowest storage cost. As we can see, the key storage costs of our scheme are higher. Considering the features provided by our scheme, the higher costs are in an acceptable range.

Table 6. Comparisons of transmission costs

Scheme	**Sender**	**Router**	**LSC**	**Member**
LKH	E_S	N/A	N/A	D_S
CS	E_A	E_A	0	D_A
SIM-KM	E_A	N/A	$E_R/0$	D_A
Sempre	$2E_S$	$E_S + D_S$	0	$2D_S$

Table 7. Comparisons of rekeying costs

Scheme	**KMC**	**Sender**	**Router**	**LSC**	**Member in the same subgroup**	**Member in other subgroups**
LKH	$O(\log_2 \mathcal{N})$	$O(1)$	N/A	N/A	$O(\log_2 \mathcal{N})$	$O(\log_2 \mathcal{N})$
CS	$O(\mathcal{H})$	0 or $O(1)$	$O(1)$	$O(\mathcal{M})$	$O(1)$	0
SIM-KM	$O(1)$	0	N/A	$O(\mathcal{M})$	$O(1)$	0
SIM-KM (periodic)	$O(1)$	$O(1)$	N/A	$O(1)$	$O(1)$	$O(1)$
Sempre	$O(1)$ or $O(\mathcal{H}) + O(\mathcal{P})$	0 or $O(1)$	$O(1)$	$O(\log_2 \mathcal{M})$	$O(\log_2 \mathcal{M})$	0
Sempre (periodic)	0	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$

5.4.2. *Transmission costs.* Here we discuss transmission costs of related schemes in terms of number of encryption/decryption. E_S and D_S represent symmetric-key encryption and decryption, respectively. And E_A and D_A represent asymmetric-key encryption and decryption, respectively. Of course the cost of one E_A is much higher than that of one E_S. E_R represents asymmetric-key based proxy re-encryption. And the cost of one E_R is usually lower than the cost of $D_A + E_A$ [11, 12]. The result is shown in Table 6. Because the KMC does not involve in the transmission process, we do not list it in the table.

LKH uses symmetric-key algorithms and uses a group key; therefore, the transmission cost of the sender is E_S and that of a member is D_S.

Reversing functions and secret functions used in CS are asymmetric-key based; therefore, the transmission cost of the sender is E_A and that of a member is D_A. Each intermediate node uses a secret function to process multicast traffic, so its cost is also E_A.

In SIM-KM, asymmetric-key based schemes are used, so the cost of the sender and a member is E_A and D_A, respectively. An LSC needs to perform proxy re-encryption during rekeying, its cost is E_R. When transformation is stopped, the cost of an LSC is 0.

In our scheme, multicast data are transmitted in the data transfer mode. So the costs discussed here are those in the data transfer mode, in which symmetric-key based algorithms are used. Because the sender encrypts multicast data twice, its cost is $2E_S$. Similarly, the cost of a member is $2D_S$. Each router decrypts and then encrypts, so its cost is $E_S + D_S$. The transmission costs of our scheme are lower than CS and SIM-KM.

5.4.3. *Rekeying costs.* The comparisons of rekeying costs are listed in Table 7. Assume the number of downstream routers connected to a router is $\mathcal{P}$. And $\mathcal{H}$ denotes the length of the chain of routers. To show the effectiveness of containment, we consider the costs of "a member in the same subgroup" and the costs of "a member in other subgroups" separately.

According to LKH's paper, RFC 2627 [5], the cost of the KMC is $O(\log_2 \mathcal{N})$. The sender only changes the group key, so its cost is $O(1)$. In LKH, a subtree could be viewed as a subgroup. Therefore, we consider members in subtrees which do not include the joining/leaving member as "members in other subgroups". But whether or not a member is in the same subgroup with the joining/leaving member, its cost is the same $O(\log_2 \mathcal{N})$.

CS also has the cases of "a chain of routers", so the rekeying cost of the KMC is $O(\mathcal{H})$. The rekeying cost of the sender is usually 0. Only when a group is newly created, the cost is $O(1)$. When a rekeying event happens, the responsible router changes its secret function, so its cost is $O(1)$. Assume an LSC is responsible for managing local subgroup keying material, it sends the new reversing function to all remaining members in the local subgroup. The cost of an LSC is $O(\mathcal{M})$, because $\mathcal{M}$ rekeying messages are sent to local members. The cost of a member in the same subgroup is $O(1)$. Members in other subgroups are not affected by rekeying events, so their costs are 0.

Because SIM-KM contains periodic rekeying, we discuss the rekeying costs of SIM-KM separately in two cases: (a) membership rekeying, which happens when a member joins or leaves, and (b) periodic rekeying, which happens periodically. First, we discuss the case (a). When a rekeying event occurs, a group controller may contact the GM, and therefore the cost of the GM is $O(1)$. The rekeying cost of the sender is 0 because the rekeying procedure is not related with the encryption key held by the sender. The GC notifies its local members by sending $\mathcal{M}$ rekeying messages, so its cost is $O(\mathcal{M})$. Moreover, the cost of a member in the same subgroup is $O(1)$, and the cost of a member in other subgroups is 0.

On the other hand, in the case (b), during a periodic rekeying event, the GM multicasts the new encryption key to all the participants and trusted controllers. Therefore, the cost of every entity in the group is $O(1)$.

Our scheme also has periodic rekeying, so we also discuss the rekeying costs of our scheme in two cases: (a) membership rekeying and, (b) periodic rekeying. In the case (a) of our scheme, if a membership change does not involve changes of routers, that is, **Procedure** `Router-Join` or **Procedure** `Router-Leave` is not executed, the KMC's rekeying cost is $O(1)$. On the other hand, when a router joins or leaves, the KMC must change $\mathcal{P}$ edge keys and $\mathcal{P}$ proxy keys. If a chain of routers also joins or leaves, the additional cost to process along the chain of routers for the KMC is $O(\mathcal{H})$. Therefore, the rekeying cost of the KMC is $O(\mathcal{H})+O(\mathcal{P})$. Network topology determines that either $O(\mathcal{H})$ or $O(\mathcal{P})$ dominates. Since both $\mathcal{P}$ and $\mathcal{H}$ are small numbers less than 100, the rekeying cost of the KMC is low. The rekeying cost of the sender is usually 0. The exception is when a new multicast tree is created, and at that time the edge key of the root edge is changed. So the cost in this case is $O(1)$. When a rekeying event occurs, a router changes its proxy keys, and the cost is $O(1)$. In our scheme, assume we use LKH as our local key management protocol, and therefore the cost of an LSC to rekey a subgroup is $O(\log_2 \mathcal{M})$. Moreover, because of using LKH, the cost of a member in the same subgroup is $O(\log_2 \mathcal{M})$. Members in other subgroups are not affected by rekeying events, so their costs are 0.

On the other hand, in the case (b), during a periodic rekeying, the new TEK is distributed by the TEK distribution mode. Therefore, the cost of every entity in the group is $O(1)$. The KMC does not know a TEK, so its cost is 0 in periodic rekeying.

Now we compare rekeying costs from views of different entities. In the aspect of the KMC's cost, LKH has the highest cost because no containment is provided. CS and our scheme have comparable costs. SIM-KM's cost of the KMC is the lowest. The costs of the sender are basically the same for all four schemes discussed. CS and our scheme grant limited trust to routers, and their costs of a router are about the same. Assume LKH is used as our local subgroup key management protocol, and it reduces the cost of an LSC at the expense of the costs of local members. Therefore, in our scheme, the cost of an LSC is lower than CS, and the cost of a member is higher. Moreover, we can clearly see the effect of containment. In two schemes with containment, CS and Sempre, members in other groups have no burden during a rekeying event. The costs of periodic rekeying of SIM-KM and Sempre are about the same. But our KMC is more efficient in periodic rekeying because it does not know a TEK. As we can see, our scheme is efficient in rekeying costs.

6. **Concluding Remarks.**

We have proposed a novel secure multicast architecture called Sempre, which is based on proxy re-encryption. The multicast model also considers routers, so it is more realistic than schemes based on logical trees. LKH-based schemes only consider trees formed by logical concepts. Because such schemes do not consider the relationship between routers and group members, they have difficulty achieving containment.

Mukherjee and Atwood's SIM-KM scheme also exploits proxy re-encryption, but we found that it fails to achieve forward secrecy. This is because the scheme uses a group key and, in the rekeying procedure, the new key is encrypted by the old key. Another drawback is that SIM-KM does not address the need for containment.

The proposed approach reduces the extent of the 1 affects n problem, which is the source of scalability problems. It also exploits proxy re-encryption to allow intermediate routers to transform ciphertext without revealing the private keys or the plaintext. By giving proper proxy keys to the intermediate routers, the impact of membership changes is confined to a local area. By extension, a security breach in a local area can also be confined to that area. Thus, we achieve the goals of containment and scalability. Moreover, since our scheme combines asymmetric-key and symmetric-key algorithms, it is practical for large and dynamic multicast groups.

For simplicity, we only discuss the KMC as a single entity; however, the task could be easily distributed among multiple entities in our architecture. Multiple KMCs eliminate the single point of failure problem, and they are more practical than a single unit. Each administrative domain has its own KMC, which is responsible for managing the keys in that domain.

Our architecture is not limited to one specific cryptographic scheme. All proxy re-encryption schemes that conform to the properties discussed in Section 5.1 can be used. Hence, under our architecture, operators have the freedom to choose different schemes. This property enhances the survivability of the whole system.

A practical example of our architecture is provided by a secure large-scale subscription scenario. In this scenario, only members who pay the subscription fee can retrieve the protected content.

A natural extension of our scheme is to allow some routers to just pass on multicast traffic without re-encryption. In the same administrative domain, several subgroups under the same upstream router could merge to form one big subgroup. Therefore, routers in the original small subgroups could just pass on multicast traffic without re-encryption. Subgroup merging reduces the transmission costs of routers, but it increases some of the costs of local members and LSCs. Moreover, a bigger subgroup means that the area affected by a security breach would be larger. Therefore, subgroup merging is a tradeoff between security and performance.

Some formal verification tools, such as Alloy [31, 32] and CORAL [33], have been used to verify secure multicast protocols. In the future, we will consider using those tools to verify our protocol.

Acknowledgment. This work is supported in part by the National Science Council under Grants NSC 99-2218-E-002-024, NSC 99-2218-E-002-026 and NSC 100-2221-E-019-045-. This work is conducted under the "III Innovative and Prospective Technologies Project" of the Institute for Information Industry, which is subsidized by the Ministry of Economy Affairs of Taiwan. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers.

REFERENCES

[1] S. Deering, Host extensions for IP multicasting, *RFC 1112*, 1989.

[2] S. E. Deering and D. R. Cheriton, Multicast routing in datagram internetworks and extended LANs, *ACM Transactions on Computer Systems*, vol.8, no.2, pp.85-110, 1990.

[3] S. Rafaeli and D. Hutchison, A survey of key management for secure group communication, *ACM Computing Surveys*, vol.35, no.3, pp.309-329, 2003.

[4] S. Mittra, Iolus: A framework for scalable secure multicasting, *Proc. of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp.277-288, 1997.

[5] D. M. Wallner, E. J. Harder and R. C. Agee, Key management for multicast: Issues and architectures, *RFC 2627*, 1999.

[6] C. K. Wong, M. Gouda and S. S. Lam, Secure group communications using key graphs, *IEEE/ACM Transactions on Networking*, vol.8, no.1, pp.16-30, 2000.

[7] Y. Kim, A. Perrig and G. Tsudik, Simple and fault-tolerant key agreement for dynamic collaborative groups, *Proc. of the 7th ACM Conference on Computer and Communications Security*, pp.235-244, 2000.

[8] R. Mukherjee and J. W. Atwood, Proxy encryptions for secure multicast key management, *Proc. of the 28th Annual IEEE International Conference on Local Computer Networks*, pp.377-384, 2003.

[9] R. Mukherjee and J. W. Atwood, SIM-KM: Scalable infrastructure for multicast key management, *Proc. of the 29th Annual IEEE International Conference on Local Computer Networks*, pp.335-342, 2004.

[10] R. Mukherjee and J. W. Atwood, Scalable solutions for secure group communications, *Computer Networks*, vol.51, no.12, pp.3525-3548, 2007.

[11] M. Blaze, G. Bleumer and M. Strauss, Divertible protocols and atomic proxy cryptography, *Proc. of Advances in Cryptology — EUROCRYPT '98: International Conference on the Theory and Application of Cryptographic Techniques, LNCS*, vol.1403, pp.127-144, 1998.

[12] A. Ivan and Y. Dodis, Proxy cryptography revisited, *Proc. of the 10th Annual Network and Distributed System Security Symposium*, 2003.

[13] G. Ateniese, K. Fu, M. Green and S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage, *ACM Transactions on Information and System Security*, vol.9, no.1, pp.1-30, 2006.

[14] M. Green and G. Ateniese, Identity-based proxy re-encryption, *Proc. of the 5th International Conference on Applied Cryptography and Network Security, LNCS*, vol.4521, pp.288-306, 2007.

[15] D. L. Cook and A. D. Keromytis, Conversion and proxy functions for symmetric key ciphers, *Proc. of the IEEE International Conference on Information Technology: Coding and Computing, Information and Security Track*, pp.662-667, 2005.

[16] D. L. Cook and A. D. Keromytis, Conversion and proxy functions for symmetric key ciphers, *Journal of Information Assurance and Security*, vol.1, no.2, pp.119-128, 2006.

[17] R. Molva and A. Pannetrat, Scalable multicast security with dynamic recipient groups, *ACM Transactions on Information and System Security*, vol.3, no.3, pp.136-160, 2000.

[18] C.-Y. Huang, Y.-P. Chiu, K.-T. Chen and C.-L. Lei, Secure multicast in dynamic environments, *Computer Networks*, vol.51, no.10, pp.2805-2817, 2007.

[19] J. Hur, Y. Shin and H. Yoon, Decentralized group key management for dynamic networks using proxy cryptography, *Proc. of the 3rd ACM Workshop on QoS and Security for Wireless and Mobile Networks*, pp.123-129, 2007.

[20] J. Y. Hwang, J. Y. Chun and D. H. Lee, Weaknesses in the Hur-Shin-Yoon decentralized group key management, *Wireless Communications & Mobile Computing*, vol.9, no.12, pp.1565-1571, 2009.

[21] A. M. Eskicioglu and M. R. Eskicioglu, Multicast security using key graphs and secret sharing, *Proc. of the Joint International Conference on Wireless LANs and Home Networks and Networking*, pp.228-241, 2002.

[22] H. Um and E. J. Delp, A secure group key management scheme for wireless cellular networks, *The 3rd International Conference on Information Technology: New Generations*, pp.414-419, 2006.

[23] H. Um and E. J. Delp, A secure group key management scheme for wireless cellular systems, *International Journal of Network Security*, vol.6, no.1, pp.40-52, 2008.

[24] D. Waitzman, C. Partridge and S. Deering, Distance vector multicast routing protocol, *RFC 1075*, 1988.

[25] J. Moy, Multicast extensions to OSPF, *RFC 1584*, 1994.

[26] D. Kosiur, *IP Multicasting: The Complete Guide to Interactive Corporate Networks*, Wiley Computer Publishing, 1998.

[27] Y.-P. Chiu, C.-L. Lei and C.-Y. Huang, Secure multicast using proxy encryption, *Proc. of the 7th International Conference on Information and Communications Security, LNCS*, vol.3783, pp.280-290, 2005.

[28] B. Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons, 1996.

[29] E. Barker, W. Barker, W. Burr, W. Polk and M. Smid, Recommendation for key management – Part 1: General, *NIST Special Publication 800-57*, 2007.

[30] S. Bradner, Key words for use in RFCs to indicate requirement levels, *RFC 2119*, 1997.

[31] D. Jackson, Alloy: A lightweight object modelling notation, *ACM Transactions on Software Engineering and Methodology*, vol.11, no.2, pp.256-290, 2002.

[32] M. Taghdiri and D. Jackson, A lightweight formal analysis of a multicast key management scheme, *Formal Techniques for Networked and Distributed Systems — FORTE 2003, LNCS*, vol.2767, pp.240-256, 2003.

[33] G. Steel and A. Bundy, Attacking group multicast key management protocols using CORAL, *Electronic Notes in Theoretical Computer Science*, vol.125, no.1, pp.125-144, 2005.