Secure Multicast Using Proxy Encryption*

Yun-Peng Chiu, Chin-Laung Lei, and Chun-Ying Huang

Department of Electrical Engineering,
National Taiwan University
{frank, huangant}@fractal.ee.ntu.edu.tw, lei@cc.ee.ntu.edu.tw

Abstract. In a secure multicast communication environment, only valid members belong to the multicast group could decrypt the data. In many previous researches, there is one "group key" shared by all group members. However, this incurs the so-called "1 affects n problem," that is, an action of one member affects the whole group. We believe this is the source of scalability problems. Moreover, from the administrative perspective, it is desired to confine the impacts of changing membership events in a local area. In this paper, we propose a new secure multicast architecture without using a group key. We exploit a cryptographic primitive "proxy encryption." It allows routers to convert a ciphertext encrypted under a key to a ciphertext encrypted under another key, without revealing the secret key and the plaintext. By giving proper keys to intermediate routers, routers could provide separation between subgroups. Therefore the goals of scalability and containment are achieved.

Keywords: Secure multicast, multicast key management, cipher sequences, proxy encryption, ElGamal cryptosystem.

1 Introduction

Since the commence of multicast communications in the late 1980s [1], secure multicast communication have been frequently addressed in the literature. Rafaeli and Hutchison's paper [2] provided a detailed survey on secure multicast.

Quite a few researches in this area made use of a group key, which is shared among all group members. The sender encrypts the multicast data using this group key, and all valid members use the same group key to decrypt. However, the existence of this group-wise key incurs the so-called "1 affects n problem" [3], which means an action of one member affects the whole group. More specifically, since the group key is known by all members, whenever a member joins or leaves the group, everyone remains in the group must acquire a new group key.

To build a practical secure multicast architecture, we focus on scalability and containment issues. Scalability means that the processing overhead of each security action should be minimized in terms of number of group members. Containment means that a security event occurs in one subgroup does not affect other subgroups.

* This work was supported in part by Taiwan Information Security Center (TWISC), National Science Council under the grants NSC 94-3114-P-001-001-Y.

S. Qing et al. (Eds.): ICICS 2005, LNCS 3783, pp. 280–290, 2005.

To aim at the above issues, we adopt two techniques. First, distribute the computation loads to intermediate routers. This makes the whole architecture scalable. And second, make a dependency between keying material and the topology of the multicast network. This dependency assures the containment of security exposures.

A naive method to provide containment is to decrypt and then encrypt again at intermediate routers [3]. This method requires fully trust to routers because routers have the ability to decrypt the plaintext, which is an undesirable feature.

In this paper, we propose a new secure multicast architecture for large and dynamic groups. Specifically, we focus on the one-to-many communication pattern. We exploit a cryptographic primitive "proxy encryption." By using this primitive, a proxy (router) could convert the ciphertext for one person into the ciphertext for another person without revealing secret decryption keys or the plaintext. Therefore the goal of scalability and containment could be achieved.

The rest of this paper is organized as follows. Related works and the basic concept of proxy encryption are discussed in Section 2. Section 3 describes the proposed secure multicast architecture based on proxy encryption. We analyze the proposed scheme, and compare it with related works in Section 4. Finally, Section 5 concludes this paper.

2 Related Works

In this section, we discuss some previous researches. Logical Key Hierarchy (LKH) may be the most representative research in this area; many researches followed their methodology and tried to enhance it. The cipher sequences framework (CS) tries to solve the multicast security problem using a different methodology. The most important advantage of CS is the containment. We also discuss Mukherjee and Atwood's researches, which also make use of proxy encryption.

2.1 Logical Key Hierarchy

Logical Key Hierarchy (LKH) is separately proposed by Wallner et al. [4] and Wong et al. [5]. 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 Fig. 1, member u_5 will have k_5, k_{56}, k_{58}, and the group key, k. Therefore, in a balanced tree, a member will have $(\log_2 \mathcal{N})+1$ keys, where $\mathcal{N}$ is the group size, and $\log_2 \mathcal{N}$ is the height of the tree. When a rekeying is needed, these KEKs could be used to encrypt new KEKs. For example, if member u_5 leaves the group, we must change those keys it knows. 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 to remaining members. We encrypt new k'_{56} using k_6, and encrypt new k'_{58} using k'_{56} and k_{78}, respectively. Then k' is encrypted using k'_{58} and k_{14}, respectively.

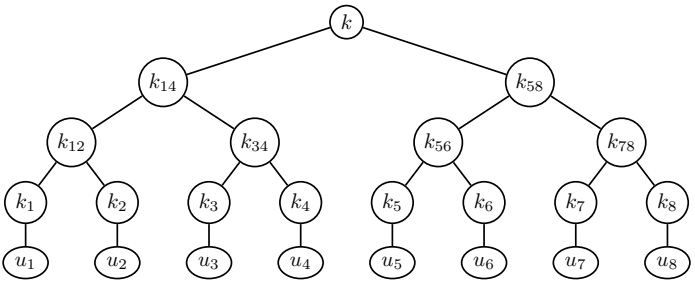

Fig. 1. An LKH tree

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.

2.2 Cipher Sequences

The cipher sequences framework (CS) was proposed by Molva and Pannetrat [6]. By distributing secure functions to intermediate nodes, keying material has a dependency on the multicast network topology. Therefore the containment of security exposures is assured.

Assume S_0 is the information to be multicast. Each node N_i is assigned a secret function f_i. N_i receives multicast data from its parent node N_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 can use h_i to get the original multicast data by calculating: $S_0 = h_i(S_n^i)$.

For example, Fig. 2 depicts a simple tree with five cipher sequences. We follow one cipher sequence from the root to the leaf M_5. First, the root computes $f_1(S_0)$ and sends the result to its children inner nodes. 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 the leaf M_5. Finally, the leaf M_4 receives $S_3^4 = f_6(S_2^4)$ and recovers the original multicast data by computing $S_0 = h_4(S_3^4)$.

2.3 Proxy Encryption

Proxy encryption was first introduced by Blaze, Bleumer, and Strauss in 1998 [7]. 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.

In traditional public-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. On the contrary, in a proxy encryption scheme, a new role, proxy P, is introduced. P is given a proxy key $\pi_{A \to B}$, and P could convert a ciphertext originally for user A to a message which could be decrypted using user B's private key. P could not decrypt the ciphertext, nor gain information about A's or B's private keys.

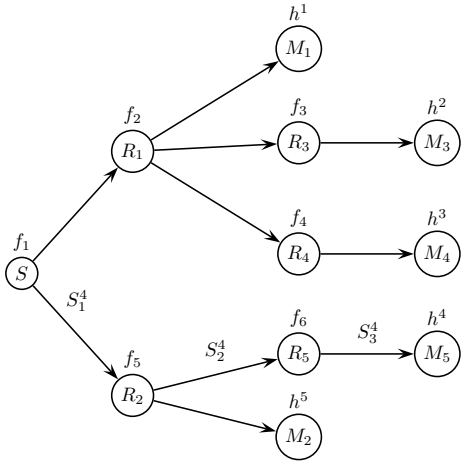

Fig. 2. An example of CS tree

In this paper, we use the "unidirectional ElGamal encryption scheme" proposed by Ivan and Dodis [8] as an example of underlying proxy encryption scheme in our architecture. Our architecture is not limited to it, and other proxy encryption schemes could also be used.

Assume an ElGamal encryption scheme is defined as a three-tuple: (KeyGen, Enc, Dec). The key generation algorithm KeyGen outputs a public key $\mathsf{EK} = (g, p, q, y)$ and a private key $\mathsf{DK} = (x)$, where q is a prime number, p is a prime number of the form $2q + 1$, g is a generator in $\mathbb{Z}_p^*$, x is randomly chosen from $\mathbb{Z}_q$, and $y = g^x \bmod p$. The encryption algorithm Enc is defined as $c = \mathsf{Enc}(m, \mathsf{EK}) = (g^r \bmod p, mg^{xr} \bmod p)$, where r is randomly chosen from $\mathbb{Z}_q$. r is chosen by the sender, and r should be used only once. The decryption algorithm Dec decrypts the ciphertext by computing $mg^{xr}/(g^r)^x = m \pmod p$.

The unidirectional ElGamal encryption scheme proposed in [8] could be defined as (KeyGen, Enc, Dec, ProxyKeyGen, ProxyEnc). The key generation algorithm KeyGen outputs a public/private key pair as the original ElGamal encryption scheme. The encryption and decryption algorithms are also the same with the original version. ProxyKeyGen splits $\mathsf{DK}_A = x$ into two parts x_1 and x_2, where $x = x_1 + x_2$. Then x_1 is given to the proxy, and x_2 is given to user B. Using ProxyEnc, P could convert a message originally for user A to a message for user B. ProxyEnc has two parameters: c and $\pi_{A\to B}$, here $c = \mathsf{Enc}(m, \mathsf{EK}_A)$, and $\pi_{A\to B} = x_1$. On receiving $c = (g^r \bmod p, mg^{xr} \bmod p)$, the proxy computes $mg^{xr}/(g^r)^{x_1} = mg^{(x-x_1)r} = mg^{x_2 r}$, and then sends $(g^r \bmod p, mg^{x_2 r} \bmod p)$ to user B. Finally, user B can decrypt this converted message as $mg^{x_2 r}/(g^r)^{x_2} = m$.

2.4 Mukherjee and Atwood's Researches

Mukherjee and Atwood proposed a key management scheme exploiting the proxy encryption technique in [9]. In their scheme, there are 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 the encryption/decryption keys. In a multicast tree, there may be several GCs, and each GC is associated with a subtree of the distribution tree. GCs perform key management functions and transform the ciphertext using proxy encryption. Participants join the GC nearest to them, and get keys from the GC.

When a rekeying is required, the GC "splits" the group decryption key to the "proxy encryption key" and the "proxy decryption key." For example, in the unidirecional 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.

When a member joins, the GC sends the proxy decryption key to the joining participant over a secure channel protected using 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: unicasting the proxy decryption key to each participant over separate secure channels, or, encrypting the proxy decryption key using each participant's K_g and multicasting an aggregated message.

In this scheme, a group key is still used. Proxy encryption is used only between a membership change event and a periodic rekeying. After a periodical rekeying, a new group key is used and the transformation is stopped. On the contrary, in our scheme, proxy encryption is always used to transmit data. Moreover, in their scheme, intuitive methods are used to to deliver the proxy decryption key to participants, which brings a large burden to GCs.

3 The Proposed Scheme

In this section, we propose a new secure multicast architecture making use of the proxy encryption mechanism discussed in the previous section. In the following subsections, we first extend the unidirectional ElGamal encryption to a proxy sequence. Then our multicast model is described in Section 3.2. Finally we extend the unidirectional ElGamal encryption to a multicast tree in Section 3.3.

3.1 Extend to a Proxy Sequence

In this subsection, we extend the original unidirectional ElGamal encryption scheme to a case with a sequence of proxies, as shown in Fig. 3. The sender S sends the message along the path. Routers R_1 and R_2 play the role of proxies. The final destination is M.

Here we introduce a new concept "segment key." Each link between two nodes is called a "segment." Each segment is assigned a public/private key pair by a trusted server (TS). These segment keys are actually stored in TS. Moreover, TS calculates proxy keys according to segment keys, and distributes proxy keys to intermediate routers by secure channels.

We show an example of using the unidirectional ElGamal encryption scheme in Fig. 3. All parameters are the same with those in the unidirectional ElGamal

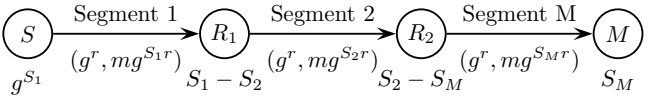

Fig. 3. Proxy sequence using unidirectional ElGamal encryption

encryption scheme. We let the proxy key between Segment A and Segment B be the difference of their corresponding private key, i.e., $\pi_{A \to B} = A - B$, where A and B are private keys of Segment A and Segment B, respectively. At first, S encrypts message m using Segment 1's public key, g^{S_1}, and sends the ciphertext $(g^r \bmod p, mg^{S_1 r} \bmod p)$ to R_1. TS gives R_1 a proxy key, $\pi_{S_1 \to S_2} = (S_1 - S_2)$, therefore R_1 computes $mg^{S_1 r}/(g^r)^{(S_1 - S_2)} = mg^{S_2 r}$, and then sends $(g^r \bmod p, mg^{S_2 r} \bmod p)$ to R_2. Similarly, R_2 has $\pi_{S_2 \to S_M} = (S_2 - S_M)$, so it converts the received ciphertext into $(g^r \bmod p, mg^{S_M r} \bmod p)$. Finally, M has S_M, therefore it could compute $mg^{S_M r}/(g^r)^{S_M} = m$ to decrypt the ciphertext.

3.2 Multicast Model

Our multicast model is similar to that in the cipher sequences framework. We use Fig. 2 again to describe our multicast model. In a multicast routing protocol, routers form a multicast tree to transmit multicast traffic. The root S is the source, intermediate nodes R_i, where i is an integer, are routers, and every leaf node represents a set of local subgroup members attached to the same router. M_i is the set of local subgroup members attached to R_i. Each router may have local subgroup members and/or downstream routers. Note in the cipher sequence framework, intermediate routers do not have local members.

In LKH schemes, all keying operations are done by senders and members; routers are not involved. Instead, in our scheme, we make use of routers, because this reduces the loads of senders and members. The trade-off is that we must grant some trust to routers. We assume routers faithfully transfer and convert encrypted multicast data.

3.3 Extend to a Multicast Tree

Based on Section 3.1, now we further extend to a multicast tree, for example as shown in Fig. 4. We assume that TS knows the overall topology of the multicast tree. Although TS seems to be centralized, we think the tasks of TS could be easily distributed over several network entities. As mentioned, TS generates keys and distributes them to intermediate routers by secure channels.

We define a term "downstream-segment set," which is a set includes segments among a given router and all its downstream routers. (Note the segment between a router and its local subgroup is not included in this set.) For instance, in Fig. 4, Segment 3 and Segment 4 are in the same downstream-segment set. In our scheme, segments belong to the same downstream-segment set are given the same segment key.

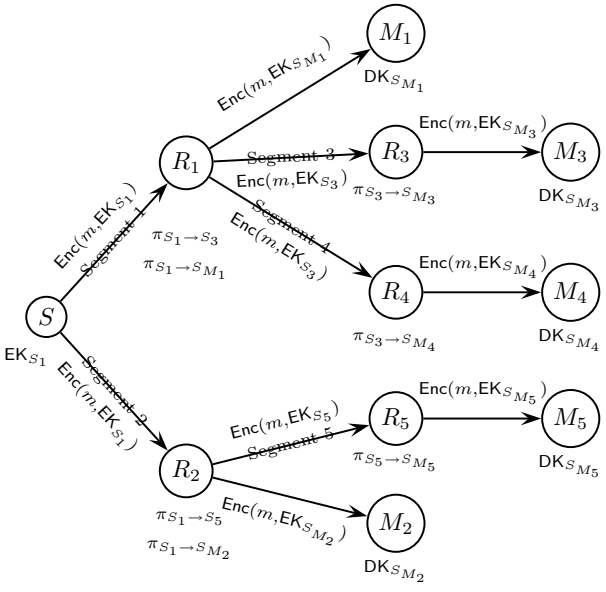

Fig. 4. Proxy tree

Each intermediate router is given two keys: the *downstream routers conversion key* and the *local subgroup conversion key*. The former is used to convert the upstream ciphertext to another ciphertext for downstream routers; it is calculated according to the upstream segment's key and the downstream segment's key. On the other hand, the latter is used to convert the upstream ciphertext for its local subgroup members only; it is calculated according to the upstream segment's key and the local subgroup key.

Each local subgroup has its own local subgroup key. Members of local subgroup use this key to decrypt the ciphertext from the attached router. We assume each local subgroup uses its own secure multicast key distribution protocol. This permits local policy of choosing a key distribution protocol. Every subgroup could choose its own key distribution protocol independent from other subgroups. Since the size of a local subgroup is much smaller than that of the whole group, using LKH is acceptable in local subgroups.

We also assume every subgroup has a "subgroup controller," who is responsible for local subgroup key management. It maintains logical keys for the local subgroup, and securely transmits the local subgroup key to TS. After members of local subgroup have determined their subgroup key, the subgroup controller securely transports their subgroup key to TS, then TS computes the proxy key for the attached router according to the local subgroup key.

The key assignment algorithm is described in Algorithm 1. We use Fig. 4 as an example. S encrypts the message m using Segment 1's public key, EK_{S_1}, and sends the ciphertext $\mathsf{Enc}(m, S_1)$ to both R_1 and R_2. We send the same encrypted message to R_1 and R_2, and this could preserve the benefit of multicast: the same

Algorithm 1: Key assignment algorithm

```
Input: Multicast tree topology
Output: Key assignment on every router
for every segment except that between a router and its local subgroup do
    if no segment in the same downstream-segment set is already given a key then
        assign a segment key;
    end
end
for every router do
    if has downstream routers then
        assign a downstream routers conversion key according to the upstream segment's
        key and the downstream segment's key;
    end
    if has local members then
        assign a local subgroup conversion key according to the upstream segment's key
        and the local subgroup key;
    end
end
```

data is sent to different paths. R_1 is given $\pi_{S_1 \to S_3}$, then it sends $\mathsf{Enc}(m, \mathsf{EK}_{S_3})$ to R_3 and R_4. R_1 also has $\pi_{S_1 \to S_{M_1}}$, this proxy key allows R_1 to convert the upstream ciphertext into $\mathsf{Enc}(m, \mathsf{EK}_{S_{M_1}})$, which could be decrypted only by using M_1's local subgroup key, $\mathsf{DK}_{S_{M_1}}$. Similarly, R_4 is given $\pi_{S_3 \to S_{M_4}}$, therefore it converts $\mathsf{Enc}(m, \mathsf{EK}_{S_3})$ to $\mathsf{Enc}(m, \mathsf{EK}_{S_{M_4}})$. M_4 could use its local subgroup key DK_{M_4} to decrypt the message.

In this way, every intermediate router converts the upstream ciphertext into the downstream ciphertext or the local subgroup ciphertext. On each link, multicast data is encrypted, so it is infeasible for an eavesdropper with no proper keys to decrypt multicast data. Therefore, we achieve secure transmission of multicast traffic.

3.4 Rekey

When a member joins or leaves the group, the local subgroup key should be changed. At the same time, the router responsible for this subgroup should also change a new proxy key according to the new local subgroup key in order to convert ciphertext for local members use.

When there are downstream nodes/routers interesting to the group, a multicast router should connect itself (and its downstream routers) to the multicast tree of this group. On the other hand, when a router has no downstream nodes/routers belong to this group, this router will be removed from the tree. When a router joins/leaves the tree, related segment keys must be changed; routers related with these segment keys must also change new proxy keys. New keys are generated and distributed by *TS* securely.

In multicast security, forward/backward secrecy are usually discussed, we follow the definitions in [10]. Forward secrecy is that a passive adversary who knows a subset of old group keys cannot infer subsequent group keys. Backward secrecy is that a passive adversary who knows a subset of group keys cannot infer previous group keys.

Algorithm 2: Rekey when a router joins/leaves

Input: Multicast tree topology, the ID of joined/left router: R
Output: New key assignment on related router
change R's upstream segment's key;
change the segment keys in the same downstream-segment set;
change the downstream routers conversion key of routers connected to those changed segments;

Algorithm 2 describes the rekeying algorithm when a router joins or leaves. Look Fig. 4 for example, if all members in M_4 leave the group, R_4 will be deleted from the tree. In order to achieve forward secrecy, the related segment key S_3 should be changed to S_3', and the related routers R_3 and R_4 must change their proxy keys. R_3 and R_4 get new proxy keys $\pi_{S_3' \to S_{M_3}}$ and $\pi_{S_3' \to S_{M_4}}$, respectively. R_1 must also change a new proxy key $\pi_{S_1 \to S_3'}$. On the other hand, if a new router wants to attach himself to R_2, in order to preserve backward secrecy, Segment 5 must be rekeyed. Thus R_2's $\pi_{S_1 \to S_5}$ and R_5's $\pi_{S_5 \to S_{M_5}}$ must be changed.

4 Analysis

In this section, first we examine the security of our scheme, and then we compare features and costs of related works and ours.

4.1 Security Analysis

By the definition of proxy encryption, it is infeasible for a proxy to derive traffic decryption keys with only proxy keys. Therefore intermediate routers could not decrypt the multicast data.

The local subgroup changes a new key when a member leaves/joins. When a member joins, the new member cannot infer previous local subgroup keys. Thus the new member cannot decrypt previous multicast data. On the other hand, when a member leaves, only remaining subgroup members get this new key. Thus the left member cannot decrypt subsequent multicast data. When a router leaves/joins, its upstream segment changes a new key. New keys are given to related routers by TS securely, and the left/new router cannot infer those keys. Therefore the router will not be able to transform subsequent/previous multicast data. As we can see, forward/backward secrecy is 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.

4.2 Comparisons of Features

The comparisons of features are shown in Table 1. In LKH, no intermediate nodes are involved, so no containment is provided. All other three schemes share computation loads to routers, and grant limited trust on routers. Thus containment is provided in these schemes.

Table 1. Comparisons of features

	Trust to intermediate nodes	Containment
LKH	No intermediate nodes	NO
CS	Limited trust	YES
Mukherjee	Limited trust	YES
Our scheme	Limited trust	YES

4.3 Comparisons of Costs

In this subsection, we analyze various costs of our scheme, and compare with related works. We assume the size of the whole group is $\mathcal{N}$, and the size of local subgroup is $\mathcal{M}$. Furthermore, $\mathcal{N} \gg \mathcal{M}$. The number of downstream routers connected to a router is $\mathcal{P}$. Assume we use LKH as our local key distribution protocol in our architecture.

In CS, every member needs a reversing function, and the sender and every intermediate node need a secret function. This seems efficient, but in fact, the trade-off is that the central server assigns every member a reversing function. Therefore the rekeying cost of the central server is $O(\mathcal{M})$.

In Mukherjee and Atwood's scheme, each member stores two keys. One is group decryption key or proxy decryption key; the other is the key shared with its GC, K_g. A GC shares K_g with each member, so it stores $\mathcal{M}$ keys. And the sender needs one key.

In our scheme, a member requires $(\log_2 \mathcal{M} + 1)$ keys. By contrast, a member in the original LKH scheme stores $(\log_2 \mathcal{N} + 1)$ keys. Because LKH is only used in local subgroups, and $\mathcal{N} \gg \mathcal{M}$, members in our scheme store fewer keys than those in the original LKH schemes. Moreover, in our scheme, the sender only stores one encryption key, and intermediate routers only stores one conversion key. The result is shown in the left three columns of Table 2. As we can see, our scheme is efficient in storage.

Table 2. Comparisons of costs

	Member storage	Sender storage	Intermediate node storage	Rekeying cost
LKH	$\log_2 \mathcal{N} + 1$	$\mathcal{N}$	No intermediate nodes	$O(\log_2 \mathcal{N})$
CS	1	1	1	$O(\mathcal{M})$
Mukherjee	2	1	$\mathcal{M}$	$O(\mathcal{M})$
Our scheme	$\log_2 \mathcal{M} + 1$	1	1	$O(\log_2 \mathcal{M})$

In our scheme, because we use LKH as our local key distribution protocol, the cost to rekey a subgroup is $O(\log_2 \mathcal{M})$. When a router joins/leaves, TS must change $\mathcal{P}$ segments keys, and then change $\mathcal{P}$ proxy keys. Therefore the cost of rekeying when a router joins/leaves is $O(\mathcal{P})$. The result of comparison of rekeying cost is shown in the last column of Table 2. As we can see, our scheme is also efficient in rekeying.

5 Conclusions and Future Work

In this paper, we proposed a new secure multicast architecture. We eliminated the usage of the group key, because it is the source of scalability problems. We

exploited proxy encryption to allow intermediate routers to transform the ciphertext without revealing the secret key and the plaintext. By giving proper conversion keys to intermediate routers, the impacts of changing membership events are confined in a local area. Thus we achieved the goal of containment and scalability. Moreover, we have shown the rekeying procedure is secure and efficient. Therefore, our scheme is scalable for large and dynamic multicast groups.

Our architecture is not limited to the unidirectional ElGamal encryption scheme; other proxy encryption schemes could also be used. In the future, we will find more efficient cryptographic primitives suitable for our architecture. For example, the unidirectional identity-based encryption scheme proposed in [8] may be a good candidate. Currently, most proxy encryption schemes are for public-key cryptosystems, but there is also a research using symmetric ciphers [11]. Because of the efficiency of symmetric ciphers, that would also be a promising candidate.

References

1. Deering, S.E., Cheriton, D.R.: Multicast routing in datagram internetworks and extended LANs. ACM Transactions on Computer Systems **8** (1990) 85–110
2. Rafaeli, S., Hutchison, D.: A survey of key management for secure group communication. ACM Computing Surveys **35** (2003) 309–329
3. Mittra, S.: Iolus: A framework for scalable secure multicasting. In: Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication. (1997) 277–288
4. Wallner, D.M., Harder, E.J., Agee, R.C.: Key management for multicast: Issues and architectures. RFC 2627 (1999)
5. Wong, C.K., Gouda, M., Lam, S.S.: Secure group communications using key graphs. IEEE/ACM Transactions on Networking **8** (2000) 16–30
6. Molva, R., Pannetrat, A.: Scalable multicast security with dynamic recipient groups. ACM Transactions on Information and System Security **3** (2000) 136–160
7. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Proceedings of Advances in Cryptology - EUROCRYPT '98: International Conference on the Theory and Application of Cryptographic Techniques. Volume 1403 of LNCS. (1998) 127–144
8. Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: Proceedings of the tenth Network and Distributed System Security Symposium. (2003)
9. Mukherjee, R., Atwood, J.W.: Proxy encryptions for secure multicast key management. In: Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks. (2003) 377–384
10. Kim, Y., Perrig, A., Tsudik, G.: Simple and fault-tolerant key agreement for dynamic collaborative groups. In: Proceedings of the 7th ACM conference on Computer and communications security. (2000) 235–244
11. Cook, D.L., Keromytis, A.D.: Conversion and proxy functions for symmetric key ciphers. In: Proceedings of the IEEE International Conference on Information Technology: Coding and Computing, Information and Security Track. (2005) 662–667