An Efficient Inter-Domain Group Key Management Scheme

Pei-Ling Yu, Chin-Luang Lei, and Li-Feng Chen

Department of Electronic Engineering
National Taiwan University, Taiwan, R.O.C.
{bey, lei, solid}@fractal.ee.ntu.edu.tw

Abstract

Multicast perfectly befits many Internet applications such as teleconference, real-time information distribution, and so on. Without guarantee of security, however, usage of multicast will be limited only to academic research. One challenge of multicast security is group key management, because the overhead of multicast key management gets much heavier while the size of a group growing larger. Even worse, if membership of a group changes frequently, overhead of re-keying becomes a critical issue of group key management. In this paper, we propose a scalable and efficient group key management scheme which addresses on re-keying strategy for highly fluctuating multicast groups. We use a two-level hashing method to organize multicast key chains. Our scheme can reduce huge computation and communication overhead of re-keying when members leave.

Keywords: multicast, group key, security, hash chain

1. INTRODUCTION

Multicast is an internetworking mechanism that can deliver data stream to multiple receivers simultaneously. In recent years, multicast becomes a hot research area and attracts legion attention to the enacting of its related protocols, including aspects of multicast routing, group management, and security. The applications of multicast are widespread, such as network conferences, multi-party games, distribution of digital media (e.g., Video-on-Demand and pay per view), and distribution of real-time information (e.g., stock information, news, and traffic report).

The security issues of multicast become more and more urgent recently, because most of the applications mentioned above require some properties of confidential, authentication, and integrity. Without guarantee of security, multicast will never be applied to e-commerce world and its usage will be restricted upsettingly.

To achieve secure multicast, members of a communicating group have to share a group key so that the transmitted data can be encrypted and verified. However, the membership of the multicast groups may change frequently. For instance, while a teleconference is proceeding over Internet, someone may leave whereas someone else may join on the half way. Each time when the membership of a multicast group changes, re-keying (update of group key) has to be taken place so that new members cannot reveal old data stream and ex-members cannot decrypt transmitting data any more. In traditional group key management schemes, vast computation power and communication bandwidth has to be consumed in order to maintain frequent re-keying. If the membership of a group changes frequently, re-keying becomes the major burden of group key management.

Consider the communicating group as shown in Figure 1, the KDC (key distribution center) has to share a secret (k_i) with each group member to maintain a secure channel. Whenever re-keying takes place, the KDC can encrypt the new group key with k_i and deliver it to r_i through the secure channel. For instance, when Alice wants to join the group, the KDC has to generate a new group key and deliver it to all current members including Alice respectively. Similarly, when Bob quits the group, re-keying occurs and the KDC delivers the new group key to all members except Bob. As a result, the KDC has to encrypt and send the new group key for N times in each re-keying, where N is the number of group members. Although the KDC could encrypt the new group key with the old group key when someone joins and reduce the number of encryption and delivery to 1, this strategy would cause broken-key propagation problem. That is, when a hacker breaks a group key, he can derive all the subsequent group keys thereafter.

Several group key management schemes have been proposed, but only a handful of them address the frequent re-keying issues of dynamic groups. Some researchers adopted tree structures into their management schemes [4,5,6,21], and hence reduced the complexity of re-keying from O(N) to O(log N). Still some others [12,13,15] divided group members into more subgroups and distributed overhead of key distribution to the agents of these subgroups. In [13], Hardjono and Cain developed a distributed architecture for group key management for inter-domain IP multicast. Hence it is a very practical proposition.

In this paper, we propose an efficient group key management scheme using two-level hashing method based on Hardjono and Cain's architecture. Our scheme can greatly reduce KDCs' and local agents' overhead of re-keying, especially in member-leaving-phase re-keying.

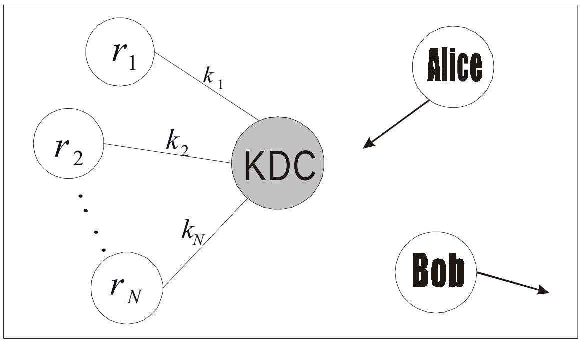

Figure 1. An example of key management for a dynamic group.

The rest of the paper is organized as follows. In the next section, Hardjono and Cain's group key management scheme is briefly introduced. Then section 3 describes our proposed scheme. Analysis of our work will be presented in section 4. Finally, conclusions are drawn in section 5.

2. PRELIMINARY

The architecture of Hardjono and Cain's scheme in [13] is shown in Figure 2. Each component is explained as follows:

AS (autonomous system) Each network domain forms an AS.

KD (key distributor) Each domain exists one or more KD servers. We call the KD in the multicast group initiator's domain as IKD (initial KD), which undertake the group key generating and distributing to other RKDs (remote KD). RKDs cooperate with IKD to send current group key to members in their domain.

LBR (leaf border router) There should be at least one border routers which are able to hand multicast traffic in each domain.

TBR (trunk border router) Each LBR is directly connected to a multicast-aware TBR in the trunk region.

2.1 New Multicast Group Initiation

Let AS1 be an autonomous system and s be the multicast group initiator in AS1. The establishment process of a multicast group is as follows:

Step N1: s notifies its LBR about the creation of a new multicast group.

Step N2: s notifies the IKD about the creation and requires it to manage the group key for the multicast group.

Step N3: IKD generates a multicast-key K_m and also generates a subgroup-key K_{AS1} for the iinitiator's AS.

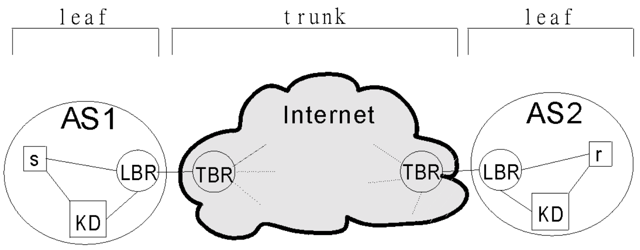

Figure 2. The architecture of group key management proposed by Hardjono and Cain.

2.2 First Member in a Remote AS

As shown in Figure 3, when the first member in a remote AS requires to join the group (say r_1 in AS2), the following steps take place.

Step F1: r_1 requests its LBR to join the multicast group.

Step F2: r_1 requests its KD to provide it with K_m. r_1 and the KD also share a secret key for on-going secure 1-to-1 communication.

Step F3: The RKD in AS2 does not have K_m yet. So it requests IKD for K_m. It also generates the sub-group key K_{AS2} for its own AS2.

Step F4: IKD sends K_m to RKD in AS2 through a secure 1-to-1 channel.

Step F5: The RKD in AS2 provides r_1 with K_m and K_{AS2}.

2.3 New Member Joining

When a host in an AS which already has at least one member (say r_2 in AS3 shown in Figure 3) wants to join the multicast group, the following steps take place.

Step J1: r_2 requests its LBR in AS3 to direct the multicast traffic to it.

Step J2: r_2 requests its KD for K_m. r_2 and the KD also share a secret key.

Step J3: The RKD issues a join-request to the IKD.

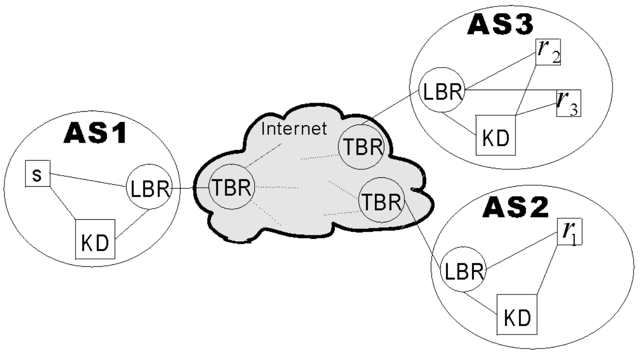

Figure 3. A snapshot of Hardjono and Cain's scheme.

$$\begin{tikzcd}
ms_0 \arrow[r, "h_1(\cdot)"] \arrow[d, "h_2(\cdot)"'] & ms_1 \arrow[r, "h_1(\cdot)"] \arrow[d, "h_2(\cdot)"'] & ms_2 \arrow[r, "h_1(\cdot)"] \arrow[d, phantom, "\vdots"] & \cdots & ms_i \arrow[d, "h_2(\cdot)"'] & \cdots \\
ss_{0,0} \arrow[d, "h_2(\cdot)"'] & ss_{1,0} \arrow[d, phantom, "\vdots"] & & & ss_{i,0} \arrow[d, "h_2(\cdot)"'] & \\
ss_{0,1} \arrow[d, "h_2(\cdot)"'] & & & & ss_{i,1} \arrow[d, "h_2(\cdot)"'] & \\
ss_{0,2} \arrow[d, phantom, "\vdots"] & & & & \vdots & \\
 & & & & ss_{i,j} &
\end{tikzcd}$$

Figure 4. Session seeds are generated by two-level hash chain. Then the group key is derived from the session seed as $mk_{i,j} = f(ss_{i,j})$.

Step J4: IKD initiates a re-keying and deliver the new group key to all existing members.

Step J5: After receiving the new group key, the RKD in AS3 sends it to r_2 through their secure 1-to-1 channel. r_2 also receives the sub-group key K_{AS3}.

In Step J4, IKD can encrypt the new multicast key with the old one and broadcast the ciphertext to all existing members. But this may suffer a propagation problem of broken keys, because someone can get all subsequent new multicast keys if he/she has broken an old one.

2.4 Member Leaving

When a member in AS2, say r_1, leaves the group, the following steps have to be executed to prevent r_1 from listening to future traffic.

Step L1: IKD generates a new multicast key and deliver it to all RKDs through 1-to-1 secure channel (unicast).

Step L2: Each RKD (except AS2) encrypts the new multicast key with its own subgroup-key and sends the encrypted multicast key to its members by broadcasting within its own domain.

Step L3: AS2 sends the new multicast key and new subgroup-key to each of its members (except the ex-member, r_1) through 1-to-1 secure channels respectively.

3. THE PROPOSED SCHEME

In this section, we present an efficient key management scheme for multicast. As in [13], we assume that all KD servers are trusted. The proposed group key management system is designed to meet the following goals.

- **Scalability**: A scalable group key management scheme should be able to serve large groups as well as sparse ones.
- **Independence**: This means that a scheme can operate well with different underlying protocols, such as cryptographic algorithms, multicast standards, and so on.
- **Security**: Only current members can decrypt multicast stream correctly.
- **Efficiency**: An efficient group key management scheme should be able to deal with highly dynamic groups.

3.1 Notations

Besides the notations introduced in the previous section, we also use the following in our scheme:

$h_1(), h_2()$: publicly known one-way hash functions.

$f()$: one-way group key generation function mapping from a session seed to a multicast group key.

ms_i: master seed, where ms_0 is chosen by IKD randomly and $ms_i = h_1(ms_{i-1})$, $i \in N$.

$ss_{i,j}$: session seed, where $ss_{i,0} = h_2(ms_i)$ and $ss_{i,j} = h_2(ss_{i,j-1})$, $j \in N$. Consequently, different master seeds can form different chains of session seeds.

$mk_{i,j}$: multicast group key, where $mk_{i,j} = f(ss_{i,j})$.

3.2 The Basic Model

Our scheme is based on [13], so the architecture and terminology is similar to [13] as shown in Figure 1. However, we adopt a two-level hashing method to generate group keys as shown in Figure 4. The IKD chooses a master seed (ms) and shares it with all RKDs. The master seed ms can generate a chain of session seeds (ss). The current session seed is shared by all existing group members. The multicast group key (mk) is derived by group key generation function $f(ss)$. Whenever a new member joins, all existing members can compute new group key by generating new session seed and then new group key. On the other hand, when a member leaves, all KDs derive a new master key and hence generate a new chain of session seeds. In this way, most burden of re-keying is distributed to local hosts and RKDs.

3.3 New Multicast Group Initiation

If someone wants to establish a multicast group (namely s in AS1 as shown in Figure 5), he has to notify his LBR and IKD and advertises just as usual multicast protocols do (e.g., IGMP [3]). IKD chooses a master seed ms_0.

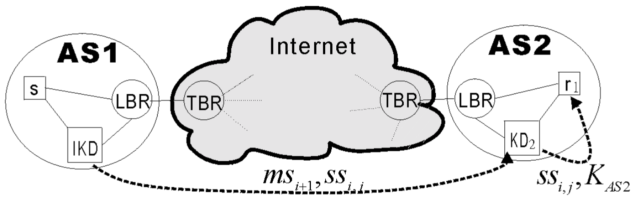

Figure 5. AS-adding-phase re-keying occurs when the first member in AS2, r_1, wants to join the multicast group.

3.4 First Member in a Remote AS (AS-Adding-Phase Re-keying)

If someone wants to join the multicast group, say r_1 in AS2 as shown in Figure 5, and he/she is the first member in that remote AS, the following steps take place:

Step a1: r_1 requests its LBR to join the multicast group.

Step a2: r_1 requests its KD (KD_2) to provide it with the multicast key.

Step a3: KD_2 informs IKD that r_1 wants to join the group.

Step a4: IKD checks the access control list. If r_1 is allowed to join, an AS-adding-phase re-keying happens and IKD informs all existing members to change multicast key. The new session seed $ss_{i,j+1} = h_2(ss_{i,j})$, where $ss_{i,j}$ is the current one, and the new multicast group key $mk_{i,j+1} = f(ss_{i,j+1})$.

Step a5: IKD sends the new session seed $ss_{i,j+1}$ and the next generation master seed ms_{i+1} to KD_2.

Step a6: KD_2 sends $ss_{i,j+1}$ and the subgroup-key K_{AS2} to r_1 through secure unicast channel. Of course, r_1 can derive valid group key hereafter.

3.5 Member Joining (Member-Join-Phase Re-keying)

This subsection describes how subsequent joiners can get into an existing subgroup (AS). For instance, r_2 in AS2 wants to join the multicast group as shown in Figure 6. There is already an existing (r_1) member in AS2. The following steps take place:

Step b1: r_2 requests its LBR to forward multicast traffic to him.

Step b2: r_2 requests its KD (KD_2) to provide him with a copy of the multicast key.

Step b3: KD_2 notifies IKD that r_2 wants to join.

Step b4: After access control checking, IKD triggers a member-join-phase rekeying.

Step b5: All existing members can generate new multicast key by computing $ss_{i,j+1} = h_2(ss_{i,j})$ and $mk_{i,j+1} = f(ss_{i,j+1})$, where $ss_{i,j}$ is the old session seed.

Step b6: KD_2 computes $ss_{i,j+1}$, and sends the new session seed and the subgroup-key K_{AS2} to r_2 through secure unicast channel.

3.6 Member Leaving

There are two different scenarios that may happen when a member leaves the group. In Case I, some member in an AS leaves and there are still other members in that subgroup. On the other hand, we will discuss removing of an AS caused by member leaving in Case II.

Case I: Member Leaving (member-leaving-phase re-keying)

When some member (namely r_2 in AS2) leaves, there may be still other hosts (r_1) in that subgroup. In such situation, the following steps must be executed.

Step c1: IKD initiates a member-leaving-phase re-keying.

Step c2: Each RKD computes the new master seed $ms_{i+1} = h_1(ms_i)$, where ms_i is the old master seed. Then they compute new session by $ss_{i+1,0} = h_2(ms_{i+1})$.

Step c3: Each RKDs (except AS2) broadcasts the new session seed to their members encrypting with their subgroup-keys.

Step c4: AS2 sends the new session seed and the new subgroup key to its current members through secure unicast channels.

After receiving the new session seed, each party can derive the new multicast group key by $mk_{i+1,0} = f(ss_{i+1,0})$.

Case II: Subgroup Removing (AS-removing-phase re-keying)

When some member (namely r_1 in AS2) leaves and there is no other member exists, the following steps must be executed.

Step d1: IKD chooses a new master seed ms_0' and sends it to all remainder RKDs through secure 1-to-1 channels.

Step d2: All KDs compute the new session seed $ss'_{0,0} = h_1(ms'_0)$.

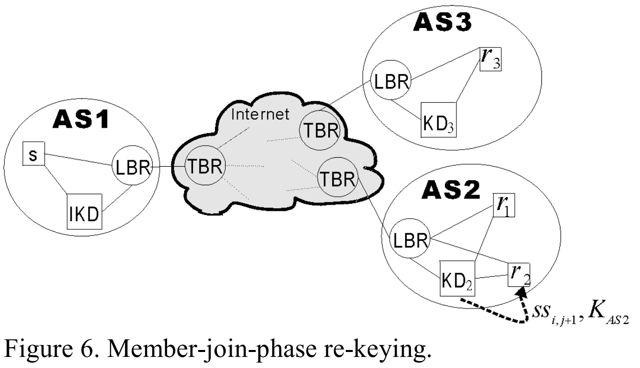

Figure 6. Member-join-phase re-keying.

Step d3: All remainder KDs broadcast the new session seed to their members encrypting with their subgroup-keys. All surviving members can derive the new group key by themselves.

4. ANALYSIS

In this section, we discuss the properties of our proposed scheme. We can prove that our scheme is secure and more efficient than previous works.

(1) New joiners can't decrypt any previous traffic

Whenever a new member joins the group, the session seed is updated by computing $ss_{i,j+1} = h_2(ss_{i,j})$. The new joiners cannot derive the old session seeds with the current session seed, because they cannot reverse the one-way hash function.

(2) Leaving hosts cannot decrypt any subsequent traffic any more

Whenever a host quits from the group, the master seed held by KDs is updated. Consequently, the session seed chain is replaced. It is useless for ex-members to possess their old session seeds. As a result, any leaving host cannot decrypt subsequent multicast packets.

(3) New RKDs cannot retrieve previous master seeds and session seeds

Whenever a RKD is added in, it is given the newest session seed and the next generation master seed. Instead of giving current master seed, IKD gives the RKD the next generation master seed. This can prevent the RKD from deriving previous session seeds ($ss_{i,1} \cdots ss_{i,j-1}$, where $ss_{i,j}$ is the current session seed). The new RKD can't derive the previous master seeds either because of one-way hash function.

(4) Removed RKDs can't derive new master seed any more

Once a RKD is kicked out, the master seed is re-chosen by IKD. As a result, the master seed which the removed RKD possesses is useless.

(5) No propagation problem of broken keys

A multicast group key is derived from a session seed by a one-way group key generation function. If someone breaks a multicast group key, he/she has no way to derive the corresponding session seed and thus cannot derive the subsequent session seeds. Therefore, our scheme does not suffer the propagation problem.

(6) Efficiency

Now, we will compare the efficiency of re-keying processes of our scheme with those of [13]. The numbers in the tables represent the amount of messages the entity has to encrypt and deliver through 1-to-1 channel. In Table 1, we can see that our scheme is a little better than [13]. However, the join phase re-keying in [13] suffers the propagation problem mentioned above. To avoid the problem, the complex must raise back to O(N).

In table 2, our proposed scheme is much more efficient than [13]. When a member leaves (but the AS is still active), the IKD need not encrypt and deliver anything. As a result, when the size of a communicating group extends and shrinks frequently, IKD can save a lot of computation and communication overhead. This may be critical because an IKD possibly serves many groups simultaneously, and each group could be very large and owns very unstable membership.

In our scheme, the only trouble situation that IKD has to deal with is AS-removing-phase re-keying. This only happens when an RKD is removed from the multicast group. Howbeit, KD servers are usually trustworthy and an empty AS probably gets some joiners soon later. Therefore IKD dose not have to kick out a RKD of an empty AS, and this could achieve better performance.

Table 1. Comparison of join-phase re-keying.

	First member in a remote AS (AS adding)			Subsequent member joining		
	IKD	RKD (new joiner's)	RKDs (others)	IKD	RKD (new joiner's)	RKDs (others)
[13]	1	1	1	1	1	1
Our scheme	1	1	1	0	1	0

Table 2. Comparison of leaving-phase re-keying.

	Member leaving (there're still others in that AS)			Member leaving (AS removing)		
	IKD	RKD (leaving member's)	RKDs (others)	IKD	RKD (leaving member's)	RKDs (others)
[13]	m	n	1	n	X	1
Our scheme	0	n	1	n	X	1

※ n is the number of RKDs; m is the number of members in a specific AS.

5. CONCLUSION

We have proposed an efficient group key management for 1-to-N inter-domain IP multicast in this paper. Our work can also be applied to N-to-N multicast with IKD being as the centralized key management server. The proposed scheme can not only reduce computation overhead of encryption but also save network bandwidth of key distribution. With our scheme, the re-keying overhead is not only reduced but also distributed, so the scheme is quite scalable and practical.

6. REFERENCES

[1] A. Ballardie, "Scalable Multicast Key Distribution" RFC 1949, May 1996.

[2] M. Burmester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System," in Advances in Cryptology – Proceedings of Eurocrypt'94, pp.275-286, 1994.

[3] B. Cain, S. Deering, and A. Thyagarajan, " Internet Group Management Protocol version 3," Internet-draft, IETF, November 1999. Available on http://search.ietf.org/internet-drafts/draft-ieft-idmr-igmp-v3-02.txt.

[4] R. Canetti, J. Garay et al, "Multicast Security: A taxonomy and Some Efficient Constructions," Proceedings of INFOCOM '99, pp. 708 -716 vol.2 1999.

[5] G. Caronni, M. Waldvogel et al, "Efficient Security for Large and Dynamic Multicast Groups," WETICE'98, pp376-383, 1998.

[6] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key Management for Secure Internet Multicast using Boolean Function Minimization Techniques," IEEE INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings Volume: 2, pp. 689-698, 1999.

[7] S.-C. Chuang, "A Flexible and Secure Multicast Architecture for ATM Networks," GLOBECOM'95, IEEE, pp. 701 -707 vol.1, 1995.

[8] S. Deering, "Host Extensions for IP Multicasting," RFC 1112, IETF, 1989.

[9] K. Djahandari and D. Sterne, "An Mbone Proxy for an Application Gateway Firewall," IEEE Symposium on Security and Privacy, pp. 72 –81, 1997.

[10] L. Dondeti, S. Mukherjee, and A. Samal, "A Dual Encryption Protocol for Scalable Secure Multicast," IEEE International Symposium on Computers and Communications, pp. 2-8, 1999.

[11] L. Gong and N. Shacham, "Multicast Security and its extension to a mobile enviroment," Wireless Networks, vol 1, pp. 281-295, 1995.

[12] T. Hardjono, "Key Management Framework for IP Multicast Infrastructure Security," IS&N'99, LNCS 1597, pp. 385-397, 1999.

[13] T. Hardjono and B. Cain, "Secure and Scalable Inter-Domain Group Key Management for N-to-N Multicast," Proceedings of International Conference on Parallel and Distributed Systems, pp.478-485, 1998

[14] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) specification," RFC 2093, IETF, July 1997.

[15] S. Mittra, "The Iolus Framework for Scalable Secure Multicasting," in Proceedings of ACM SIGCOMM'97, pp.277-288, 1997.

[16] R. Molva, Alain Pannetrat, "Scable Multicast Security in Dynamic Groups," in Proceedings of the 6th ACM Conference on Computer and Communications Security, pp101-112, 1999.

[17] R. Shankaran, V. Varadharajan, and M. Hitchens, "Secure Multicast Extensions for Mobile Networks," LCN '99. Conference on Local Computer Networks, pp. 106-116, 1999.

[18] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication," in Proceedings of the 3rd ACM Conference on Computer and Communications Security, March 1996.

[19] Debby M. Wallner, Eric J. Harder, Ryan C. Agee, "Key Management for Multicast: Issues and Architectures," Informational RFC, draft-wallner-key-arch-00.txt, July 1997.

[20] C.-K. Wong and S. Lam, "Digital Signatures for Flows and Multicasts," Sixth International Conference on Network Protocols, pp. 198 –209, 1998.

[21] C. K. Wong, M. Gouda, and S. Lam, "Secure Group Communications using Key Graphs," in Proceedings of ACM SIGCOMM'98, 1998.