A Decentralized Key Management Scheme in Overlay Multicast Network

Xingfeng Guo, Xiaodong Liu, Qionghai Dai
Broadband Network & Multimedia Research Center
Graduate school at Shenzhen, Tsinghua University, 518055, Shenzhen, P. R. China

ABSTRACT

The recent growth of the World Wide Web has sparked new research into using the Internet for novel types of group communication, like multiparty videoconferencing and real-time streaming. Multicast has the potential to be very useful, but it suffers from many problems like security. To achieve secure multicast communications, key management is one of the most critical problems. So far, a lot of multicast key management schemes have been proposed and most of them are centralized, which have the problem of "one point failure" and that the group controller is the bottleneck of the group. In order to solve these two problems, we propose a Decentralized Key Management Scheme (DKMS), using RSA key system as auxiliary keys. We analyze this scheme and find it has appropriate performance in security and scalability.

1. Introduction

The emergence of multicast makes it possible to distribute and store content in the large scope mode. Doctor S. E. Deering[1] first proposed the concept of IP multicast in 1980's, and then the multicast has had obvious progress. To support group communication, IP multicast is superior to unicast and broadcast in that it allows transmission and routing of packets to multiple destinations using fewer network resources. However, the development of IP multicast has been limited and sparse due to a variety of technical and non-technical reasons. Therefore some researchers in the resent past have proposed application layer multicast [2] as an alternate technique for multicasting. In application layer multicast, the multicasting functionality is implemented at the end hosts instead of the network routers.

To apply multicast, security is one of the essential issues that should be deliberately and carefully considered. Most of the multicast security schemes mainly use cryptographic technique, cryptographic keys should be managed in a way that can achieve secure and efficient multicast communications. So far, a set of multicast key management schemes have been proposed and most of them are centralized, which has the problem of "one point failure" and has the bottleneck at the group controller. In order to solve these problems, we propose a Decentralized Key Management Scheme (DKMS), using RSA key system as auxiliary keys. Experiment result shows that this scheme has appropriate performance in security and scalability.

This paper is organized as follows: Section 2 presents some centralized key management schemes; Section 3 introduces the decentralized key management scheme; We compare the scalable performance of the DKMS and the centralized schemes and present the experiment results in Section 4, Section 5 concludes the paper.

2. Centralized Key Management Schemes

In this section, we present two representative centralized key management schemes.

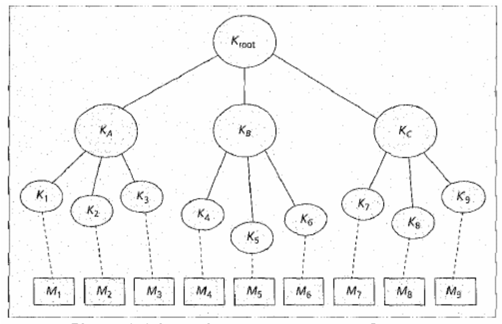


Figure 1. A k-ary key manegement tree for $k = 3, n = 9$

Figure 1 shows a tree-based key management scheme [3]. A multicast group has n members, M_1 through M_n, and a centralized group controller. The controller stores a k-ary tree structure in which each node contains a key. At the leaves of tree are n secret keys. Each member in the tree stores a subset of the controller's keys. The total number of keys stored by the controller is approximately $(kn-1)/(k-1)$, and the total number of keys stored by each member is $\log_k n$. When the group's membership changes, all keys known by the changed member must be updated. In general, for a k-ary tree we must update keys at $\log_k n$ levels in the tree, and at each level we must send k update messages. Thus, the tree-based key management scheme can update keys using $k\log_k n$ messages.

There are several important points to note about this scheme. First, it requires a reliable multicast infra-structure. This is necessary because if a group member misses a key update, he will not be able to participate in the group communication or decrypt future key update messages. Second, this scheme has the problem of "one point failure". The controller is so important that if the controller has an error the whole group members can't work anymore. Besides this, the controller's capability is the bottle-neck of the group's performance obviously.

The other example for centralized key management is the scheme based on "linear array" [4]. This scheme is not practical, because it can be broken under certain circumstances. It is interesting, however, for the insight it provides into the group key management problem.

A multicast group has n members, and a group controller. According to n members, there are n foreward keys f_i $1 \le i \le n$ and n backward keys b_i $1 \le i \le n$. The member M_i has a set of foreward keys $FSet(u_i)=\{f_j | 1 \le j \le i\}$, and a set of backward keys $BSet(u_i)=\{b_j | i \le j \le N\}$. We use GK for the present group key, GK' for the new group key, SKi for the transfer key of member i, G for active member, KC for the key controller, and "$\Rightarrow$" for multicast, "$\rightarrow$" for unicast, "{}" for encryption, so when the group members are changed, the scheme works as follows:

Member joins

$$KC \Rightarrow G : \{GK'\}GK$$

$$KC \rightarrow M_i : \{GK'\}SK_i$$

Member leaves

$$KC \Rightarrow G : \{\{GK'\}f_{i+1}\}GK \ \text{ if } i < n$$

$$KC \Rightarrow G : \{\{GK'\}b_{i-1}\}GK \ \text{ if } i > 1$$

This scheme has great performance in scalability. But it is vulnerable to collusion, any two members' collusion will harm the whole group's security. This scheme also has the problem of "one point failure" and "bottleneck" like the "tree-based" scheme has.

3. Decentralized Key Management Scheme(DKMS)

As we mentioned above, most key management schemes are centralized. It is easy to manage the keys, but the key controller is so important that once the controller is in failure the whole group can't work. The key controller is the bottle-neck of the group in capability also. In order to solve these problems, we propose a Decentralized Key Management Schemes (DKMS), using RSA key system as the auxiliary keys.

3.1 Group's Topology

In the centralized key management scheme, the group controller has the responsibility for the key management. There is no such point in DKMS, so the performance of key management is based on the group's topology obviously.

Generally, there are two layers of topology in a multicast group, the control topology and the transfer topology. The DT [5] overlay has good characters for the control topology. First, every node in the DT overlay has six neighbors at most in general. And then, one node's change won't cause topology's change in large scope. These two characters are good for scalability, and we generate our control topology with the DT algorithm.

For the transfer topology, there are many types of tree have been proposed. We compare the performance of the Shortest Path Tree (SPT) and the Rendezvous Point Tree (RPT). RPT is better than SPT in the average transfer length, but RPT has one important point like the key controller in the centralized scheme. So we choose SPT as the transfer topology.

Figure 2 shows the two layers of topology, the mesh in tint is the control topology and the tree in bold is the transfer topology.

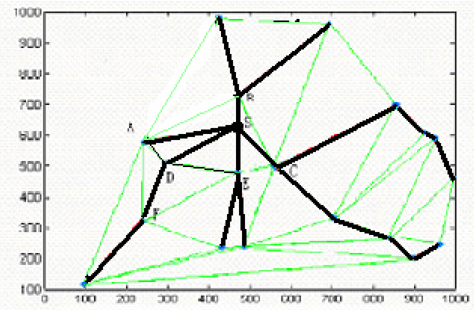


Fig. 2. control topology and transfer topology

3.2 Initialization of Keys

Two types of keys are generated in the decentralized key management scheme, the Transfer Encrypt Key (TEK) and the Key Encrypt Key (KEK). TEK is used to encrypt and authenticate the multicast channel and KEK is the auxiliary keys used to rekey the group.

TEK is generated by the root of SPT, the data source of the multicast group and can be updated periodically. For the auxiliary keys, we choose RSA keys as KEK. A member that wants to join the multicast group needs to generate its RSA keys first, and then send the RSA public key to a RSA publish server. This publish server is not a part of the multicast group, but a third party that reliable in the Internet.

All members in the SPT have a list of child-nodes and get child-nodes' RSA public key from the reliable server. In figure 2, the source member S has five children in SPT, so it contacts the publish server and gets member A,B,C,D,E's RSA public keys from the server. Member D has only one child in SPT, so it gets member F's RSA public key from the server. In the overlay constructed with DT algorithm, every node has six neighbors at most in general, so the member in SPT has six children at most in its child-nodes list.

3.3 Management Operations

This section describes how the decentralized scheme works, including TEK's distribution and TEK's update.

TEK's distribution

In the multicast group, the root of SPT first generates TEK for encryption and authentication of the multicast channel. Then the root checks its child-node list and encrypts the TEK with its children's RSA public key respectively. For example, in figure 2 the root S has five children. The root S will encrypt TEK with member A's RSA public key and send this message to member A, encrypt TEK with B's RSA public key and send this message to member B, etc. So the root S needs to encrypt five times for the TEK's distribution. The member D receives the encrypted message and decrypts this message with its RSA privacy key, so the member D could get the TEK. Then D checks its child-nodes list and finds that it has one child F in SPT, so D encrypts TEK with F's RSA public key again and sends it to the member F. In this scheme, every node in SPT needs to do decryption once and encryption according to the child-nodes list. For the DT topology's capability, the cost for key's distribution at every node is constant.

TEK's update

The multicast group is always in dynamic, and in order to insure transfer's security, TEK should be updated whenever the group member changes.

When a new member wants to join the multicast group, it first contacts the publish server and sends its RSA public key to the server. Then the new control topology and transfer topology will be generated. As we mentioned above, in the overlay constructed by DT algorithm, one node's change causes small changes to the overlay, so the re-construction will cost small. Every node in new SPT will update its child list and repurchase the RSA public keys from the publish server. After every node has updated its child list, the data source will generate the new TEK and distribute it to every node as mentioned above.

When a member wants to leave the group, the new control topology and transfer topology will be reconstructed and members in new SPT will update the child list as the member joins.

4. EXPERIMENTAL RESULTS

We do some experiments to compare the performance between the DKMS and the centralized schemes.

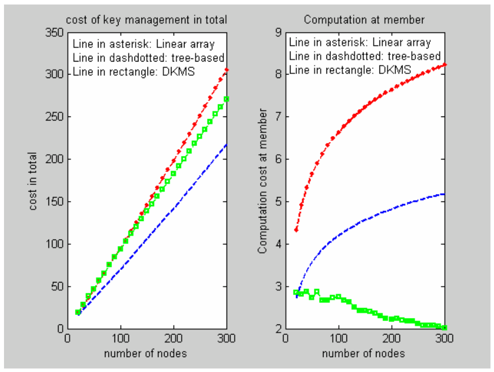
Table below outlines the basic facts about each key management scheme mentioned above (n is number of group members and k means k-ary tree).

	Linear array	Tree-based	DKMS
Centralized architecture	yes	yes	no
Keys store at controller	2n+1	$n+\frac{n}{2k}\log_k n$	null
Keys store at members	$\log_2 n$ +1	$\log_k n+1$	Constant 6 at most
Msgs sent on join/leave	2	$k\log_k n$	n
Computation at controller on join/leave	$\log_2 n$	$k\log_k n$	null
Computation at Member on join/leave	$\log_2 n$	$\log_k n$	Constant 6 at most
Vulnerable to collusion	no	yes	yes
Join security	no	yes	yes
Leave security	no	yes	yes

Figure 3 gives the experiment result about scalable performance. Figure in left shows the total cost of the key management. It consists as follows: the cost of key's store takes 5%, the cost of key's generation takes 15%, and the cost of key's update takes 80%. Figure in right shows the computation cost at member on join/leave.

5. CONCLUSION

This DKMS scheme uses DT algorithm to construct the control overlay and RSA key system as KEK. It solves two problems that centralized scheme holds, "one point failure" and the bottle neck of the group.



The theory analysis shows that this scheme has appropriate performance in security. The experiment result shows that the DKMS costs as much as the centralized scheme on key's management in whole group scope, but less in the average of every node. In this DKMS scheme, every node needs to encrypt and decrypt TEK, so application layer multicast is more suitable for this scheme than IP multicast.

6. ACKNOWLEDGEMENT

This work is supported by the key project of NSFC (No.60432030)

This work is supported by the Distinguished Young Scholars of NSFC (No.60525111)

REFERENCES

[1] "IP Multicast Technology and Application", written by Marcus Goncalves and Kitty Niles, 2001.1.

[2] "A Comparative Study of Application Layer Multicast Protocols", Suman Banerjee, Bobby Bhattacharjee.

[3] "Key management for multicast: Issues and architectures", D. Wallner, E. Harder, and R. Agee, IETF, RFC2627,1999

[4] "HySOR:Group Key Management with Collusion Scalability Tradeoffs Using a Hybrid Structuring of Receivers", Jinliang Fan, Paul Judge, Mostafa H. Ammar,

[5] "Delaunay Triangulation Using a Uniform Grid", Tsung-Pao and Les A. Piegl, University of south Florida