SIKAS: A SCALABLE DISTRIBUTED KEY MANAGEMENT SCHEME FOR DYNAMIC COLLABORATIVE GROUPS

Jiang Zhang, Jian-Guang Luo, Bin Li, Shi-Qiang Yang

Department of Computer Science and Technology, Tsinghua University, Beijing China
{zhang-jiang03,luojg03,libin98}@mails.tsinghua.edu.cn, yangshq@tsinghua.edu.cn

ABSTRACT

The increasing popularity of distributed and collaborative applications prompts the need for secure communication in collaborative groups. Some distributed collaborative key management protocols have been proposed to provide group communication privacy and data confidentiality for collaborative groups. However, most of them rekey on each member change, and the costs of group rekeying can be quite substantial for large groups with frequent membership changes. In this paper, we propose a scalable distributed key management scheme using a distributed one-way function tree named SIKAS which can significantly reduce the computation and communication costs of maintaining the group key based upon period-based group rekeying. A comparison with previous work has shown that SIKAS provides scalability and rekeying efficiency while preserving both distributed and collaborative properties.

1. INTRODUCTION

The increasing popularity of distributed and collaborative applications such as tele/video-conferencing and interactive group games prompts the need for secure communication in collaborative groups. To provide group communication privacy and data confidentiality for collaborative groups, it is essential for all members in the group to establish and share a common group key for encrypting group communication data. In order to offer both backward and forward secrecy [1], the group key has to be reestablished whenever there is any change in the group membership, which is referred to as group rekeying. To solve the problem, we need a distributed collaborative group key management scheme so that the group members can establish and update a common group key for secure and private communication.

It is important to note that the type of distributed collaborative key management protocols is very different from traditional centralized group key management protocols [2,3]. Centralized protocols rely on a centralized key server to generate and distribute the group key to the group members whenever required. However, in most distributed and collaborative applications, there is no centralized key server available. To illustrate the utility of this type of applications, consider a group of people in a peer-to-peer network having a confidential and closed meeting. Hence, a distributed collaborative key management protocol which must have two following special characteristics is desired for collaborative groups.

- Distributed property: there is no centralized key server involved.
- Collaborative property: each group member contributes an equal share towards the establishment of the common group key and gains equal control over the group.

A few distributed key management protocols have been proposed to support many-to-many secure group communication. SOT [4] can support large and dynamic groups by clustering group members to localize rekeying within a cluster. However, it relies on cluster leaders to generate cluster keys, which makes SOT not applicable for collaborative groups. DISEC [5] and TGDH [6] provide a secure, distributed and collaborative key management solution which delegates key distribution tasks evenly to all the group members, but they all rekey on each member change. Hence, the costs of group rekeying can be quite substantial for large groups with frequent membership changes. In other words, for dynamic collaborative groups, the frequency of rekeying imposes an upper limit on the scalability of key management protocols that is independent of the efficiency of an individual rekeying operation.

To handle the above problem, we propose a scalable distributed key management scheme for dynamic collaborative groups named SIKAS in this paper, which presents a period-based group rekeying approach based on distributed key management protocol using a distributed one-way function tree. Period-based rekeying decouples the frequency of rekeying from the size and membership dynamics of the group. Therefore, our scheme can significantly reduce the computation and communication costs of maintaining the group key, and can easily scale to large and dynamic collaborative groups. Moreover, our approach provides rekeying efficiency while preserving both distributed and collaborative properties.

The rest of this paper is organized as follows. Section 2 provides a dissection of SIKAS. Section 3 draws a comparison analysis with previous work. Finally, in Section 4, we conclude the paper.

2. A DISSECTION OF SIKAS

2.1. Distributed One-way Function Tree

One-way function tree has been proved to be efficient and secure in centralized key management [3]. We use a distributed one-way function tree (DOFT) to efficiently maintain the group key in a dynamic collaborative group. Each member maintains a set of keys, which are arranged in a hierarchical binary tree (shown in Figure 1). We assign a node ID v to every tree node. The node ID of the root node is set to 0. Each non-leaf node v consists of two child nodes whose node IDs are given by 2v+1 and 2v+2. Each leaf node in DOFT represents a group member U_i.

For a given node v, we associate a secret key K_v and a blinded key BK_v which is computed by applying a given one-way function to K_v. Each member generates a unique secret key for itself through a secure pseudo random number generator. The secret key of a non-leaf node v can be generated by the blinded keys of two child nodes of v using a mixing function. Mathematically, we have

$$K_v = f(BK_{2v+1}, BK_{2v+2}) = f(g(K_{2v+1}), g(K_{2v+2})) \qquad (1)$$

Here g is a one-way function, f is a mixing function.

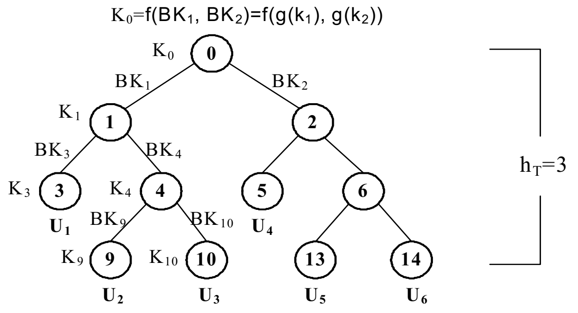

Figure 1. Distributed one-way function tree

Each member holds its own secret key and all the blinded keys of nodes that are sibling of the nodes in its key path starting from its associated leaf node up to the root node of DOFT. For a given node v, we associate an responsible member set RM_v. RM_v includes all the members in the subtree rooted at the node that is the sibling of node v. Each member computes the blinded keys of the nodes in its key path to the tree root and shares it with its associated responsible member set. For instance, as shown in Figure 1, $RM_1 = \{U_4, U_5, U_6\}$, $RM_3 = \{U_2, U_3\}$. U_1 generates BK_3 and sends it to U_2 and U_3, and generates BK_1 and sends it to U_4, U_5, and U_6. The limited distribution of secret and blinded keys is to ensure the protocol is immune to collusions.

Each member can compute the secret keys of the non-leaf nodes in its key path to the root node of DOFT. Therefore, the secret key associated with the root node is shared by all the members and is regarded as the group key. Figure 1 illustrates a possible key tree with six members U_1 to U_6. For example, U_2 holds $\{K_9, BK_{10}, BK_3, BK_2\}$. Given K_9 and BK_{10}, U_2 can generate K_4 according Eq.1. Given BK_3 and the newly generated secret key K_4, U_2 can generate K_1. Given BK_2 and the newly generated secret key K_1, U_2 can generate the secret key K_0 at the root. The secret key K_0 at root node is the group key shared by all the group members. The group key is generated in a shared and contributory fashion, so there is no single-point-of-failure.

2.2. Individual Rekeying Protocol

To provide both backward and forward secrecy, rekeying is performed whenever there is any group membership change. Let us first consider individual rekeying, meaning that rekeying is performed after every single joining or leaving event. Before the group membership changes, a special member called the sponsor is elected to be responsible for initiating the rekeying process. We use the convention that the leftmost member under the subtree rooted at the sibling of the joining and leaving nodes will take the sponsor role.

When a member leaves, all the remaining members update the structure of DOFT, and then the sponsor initiates the rekeying process. The departing member's sibling is promoted to the departing member's parent node, and the descendants of the departing member's sibling need to change their IDs. Then the sponsor changes its secret key and initiates the rekeying process. Figure 2 illustrates a member leaving case. Suppose the member U_6 leaves the group. Member U_5 becomes the sponsor. Node 13 is then promoted to node 6, and nodes 2 and 0 become renewed nodes, defined as the non-leaf nodes whose associated keys in the key tree are renewed. U_5 generates its new secret key K_6, computes the blinded keys BK_6 and BK_2. Then U_5 sends BK_6 to U_4, and sends BK_2 to U_1, U_2, and U_3. Finally, all members can compute the new group key K_0.

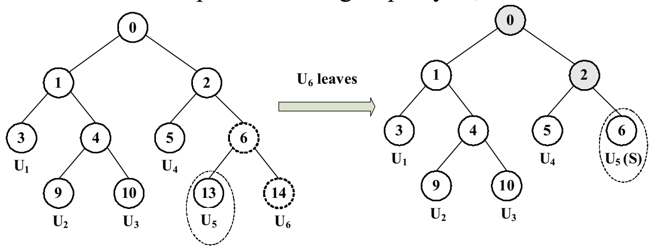

Figure 2. Single leaving case

When a new member wishes to join the group, all the existing members need to update the structure of DOFT. Each member has to first determine the insertion node under which the new member can be inserted into DOFT. We select the node of the shallowest rightmost leaf node in DOFT as the insertion node to keep DOFT as balanced as possible. To add a node v' (or tree T') to the insertion node, a new node n', is first created. Then the subtree rooted at the insertion node becomes the left child of node n', and the node v' (or the root node of T') becomes the right child of the node n'. Node n' will replace the original location of the insertion node. Then the sponsor changes its secret key and initiates the rekeying process. Figure 3 depicts a member joining case. Suppose a new member U_7 wishes to join the group. The insertion node is node 6, and the sponsor is U_5. U_7 sends its blinded key BK_{14} to U_5 upon insertion. U_5 generates a new secret key K_{13}, and computes BK_{13}, BK_6, and BK_2. Then U_5 sends BK_6 to U_4, sends BK_2 to U_1, U_2, and U_3, and sends the structure of DOFT, BK_{13}, BK_5, and BK_1 to U_7. Finally, all members can compute the new group key K_0.

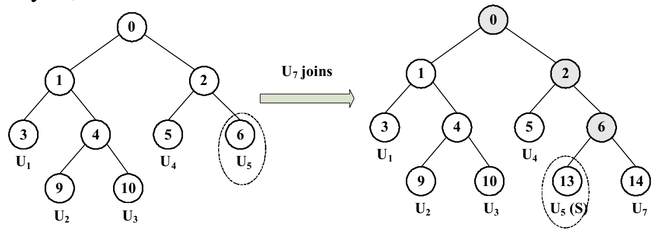

Figure 3. Single joining case

2.3. Period-based Rekeying Protocol

Based on the above leaving and joining case, we find that we can reduce one rekeying operation if we simply change the association of node 14 from U_6 to U_7. Period-based rekeying is thus proposed such that rekeying is performed on a batch of joining and leaving requests at regular period so as to reduce the number of rekeying operations. Period-based rekeying maintains the rekeying frequency regardless of the size and membership dynamics of the group, with a tradeoff of weakening both backward and forward secrecy as a result of delaying the update of the group key.

The period-based rekeying protocol is divided into two phases, namely the Pre-processing phase and the Tree-merge phase. During the current rekeying period, the Pre-processing phase is performed to pre-process all the joining and leaving requests, and all the newly joining members are appended into a temporary key tree T'. At the beginning of the next rekeying period, the Tree-merge phase is performed, and the temporary tree T' is merged to the existing key tree T. Since the period-based rekeying operations involve nodes lying on more than one key path, more than one sponsor may be elected. The paper makes the following assumptions:

- Rekeying operations of all members are synchronized to be carried out at the beginning of every rekeying period.
- The group communication satisfies view synchrony [6] that defines reliable and ordered message delivery under the same membership view.
- Each leaving or joining member broadcasts its logout or login message to all members in the group.
- All members know the existing key tree structure.

We adopt the following notations in our description. Let h_T denote the height of T, $h_{T'}$ denote the height of T'. Assume that existing $L \geq 0$ members leave and $J \geq 0$ new members join within a rekeying period. The associated leaf nodes of leaving members in DOFT form the leaving node set $V^l = \{v_1^l, ..., v_L^l\}$. The pseudo-codes of the Pre-processing phase and the Tree-merge phase are illustrated in Figure 4 and Figure 5.

```
Pre-processing ( )
1.  if (a new member joins) {
2.     if (T' = = NULL) /*no new members in T'*/
3.       create a new tree T' with the only one new member;
4.     else { /*there are new members in T'*/
5.       find the insertion node;
6.       add the new member to T';
7.       elect the leftmost member under the subtree rooted at the sibling of the joining node to be the sponsor, and initiate rekeying process;
8.     }
9.  }else if (a existing member leaves) {
10.    L= L+1;
11.    add the associated node of leaving member to $V^l$;
12. }
```

Figure 4. The Pre-processing phase

```
Tree-merge(T, T', $V^l$, L)
1.  if (L= =0) { /* there is no leave members*/
2.    if ($h_T - h_{T'} \geq 2$)
3.      select the rightmost node whose level is $h_T - h_{T'} - 1$ in T as the insertion node, then add T' to the insertion node;
4.    else
5.      select the root as the insertion node, and add T' to the root node of T;
6.  }else { /* there are leave members*/
7.    processing $V^l$, if node $v_x$ and its sibling $v_s$ are all in $V^l$, then $V^l = V^l \setminus \{v_x, v_s\}$ parent ($v_x$, $v_s$), continue the above processing until all leaving nodes are processed and $V^l$ is stable;
8.    select the node of lowest ID in $V^l$ as the insertion node;
9.    remove the subtree rooted at the remaining nodes in $V^l$ from T, and promote their siblings;
10.   add T' to the insertion node;
11. }
12.   elect members to be sponsors if they are the leftmost members of the subtree rooted at the sibling nodes of the depart leaf nodes in T, or they are the leftmost members of the subtree rooted at the sibling node of the root node of T', or they are the leftmost member of T', and initiate rekeying process.
```

Figure 5. The Tree-merge phase

In Figure 6, we show the case where some members leave and some members join during a rekeying period. Suppose new members U_8, U_9, and U_{10} wish to join the group, while U_2, U_3, U_4, and U_6 wish to leave. Then the rekeying process is as follows: (1) In the Pre-processing phase, U_8, U_9 and U_{10} first form a tree T'. Nodes 9, 21, 22, 13 are added to $V^l = \{9, 21, 22, 13\}$. (2) In the Tree-merge phase, process V^l, $V^l = \{4, 13\}$. Node 4 is elected to be the insertion node. Delete the subtree rooted at node 4 and add T' to node 4. (3) The sponsors U_1, U_7 and U_8 are elected. U_8 sends BK_4 to U_1 upon insertion. U_1 generates new K_3 and computes BK_3, BK_1. U_7 generates new K_6 and computes BK_6 and BK_2. Then U_1 sends BK_3 to U_8, U_9, and U_{10}, and sends BK_1 to U_5, and U_7. U_7 sends BK_6 to U_5, and sends BK_2 to U_1, U_8, U_9, and U_{10}. Finally, all members can compute the new group key K_0.

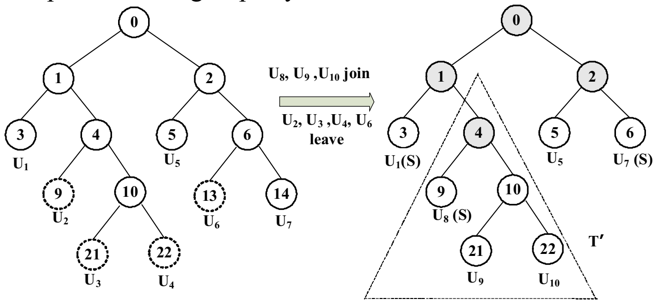

Figure 6. Period-based rekeying case

3. SCHEME ANALYSIS AND COMPARISON

In the SIKAS scheme, there is no centralized control node. Each group member contributes an equal share to the common group key, which avoids the problems with the centralized trust and the single point failure. Key distribution overhead is distributed evenly among all members. The limited distribution of secret and blinded keys can ensure the protocol is immune to collusions.

SIKAS manages keys for secure group communication using DOFT. Each member holds its own secret key and all the blinded keys of nodes that are sibling of the nodes in its key path to the root. Join/leave requires only the keys in the path from the sponsor to the root in DOFT to be changed. Thus each membership changes requires only O(log N) messages where N is the number of members in the group. Moreover, SIKAS decouples the frequency of rekeying from the size and membership dynamics of the group based upon period-based rekeying. In Pre-processing phase, SIKAS can reduce the rekeying load by preprocessing the joining members to form a temporary tree during the idle rekeying period. In Tree-merge phase, SIKAS adds the temporary tree to the shallowest node which is the root of subtree to be removed, which can keep DOFT as balanced as possible and reduce rekeying operations. As a result, SIKAS can significantly reduce the computation and communication costs of maintaining the group key in the presence of frequent membership change events.

In Table 1, we compare the analysis results of our proposed scheme (SIKAS) with those of other distributed key management schemes, such as SOT, DISEC, and TGDH. From the comparison, we can find that all the above schemes can support many-to-many secure group communication, but only the SIKAS scheme can provide scalability and rekeying efficiency while preserving both distributed and collaborative properties.

Table 1. Comparison with Other Distributed Key Management Schemes

	SIKAS	**SOT**	**DISEC**	**TGDH**
Group control	distributed	distributed	distributed	distributed
Single point of failure	No	Yes	No	No
Vulnerable to collusions	No	No	No	No
Collaborative establishment of group key	Yes	No	Yes	Yes
No. of keys in the group	O(N)	O(N)	O(N)	O(N)
No. of keys at a member	O(log N)	O(1)	O(log N)	O(log N)
Scalability	High	High	low	low

4. CONCLUSION

We propose a scalable distributed key management scheme using a distributed one-way function tree named SIKAS which can significantly reduce the computation and communication costs of maintaining the group key based upon period-based group rekeying. Our approach provides scalability and rekeying efficiency while preserving both distributed and collaborative properties.

5. REFERENCES

[1] Yacine Challal, Hamida Seba, "Group Key Management Protocols: A Novel Taxonomy", International Journal of Information Technology, 2(1), 2005, pp.105-118.

[2] C. K. Wong, M. Gouda, and S. S. Lam, "Secure Group Communication Using Key Graphs", IEEE/ACM Transactions on Networking, 8(1), 2000, pp.16-30.

[3] D.A. McGrew, A.T. Sherman. "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", IEEE Transactions on Software Engineering, 29(5), 2003, pp.444-458.

[4] W. P. Ken Yiu, S.H. Gary Chan, "SOT: Secure Overlay Tree for Application Layer Multicast", IEEE International Conference on Communications, Paris, 2004, pp.1451-1455.

[5] L. R. Dondeti, S. Mukherjee, "DISEC: A Distributed Framework for Scalable Secure Many-to-many Communication", IEEE Symposium on Computers and Communications, 2000, pp.693-698.

[6] Y.Kim, A. Perrig, and G.Tsudik, "Tree-based Group Key Agreement", ACM Trans. on Information and System Security, 7(1), 2004, pp.60-96.