Weakness of the Synchro-Difference LKH Scheme for Secure Multicast

Heeyoul Kim, Byungchun Chung, Younho Lee, Yongsu Park, and Hyunsoo Yoon

***Abstract*— Zhu proposed a Synchro-Difference LKH scheme for secure multicast that appeared in IEEE Communications Letters, vol. 9, no. 5, pp. 477–479, May 2005, which is an enhancement of the well-known LKH scheme. In this letter, we show that Zhu's scheme is insecure against collusion of two or more malicious adversaries by presenting two kinds of attack scenarios.**

***Index Terms*— Secure multicast, group secrecy, confidentiality, collusion attack.**

I. Introduction

In secure multicast, the key server (KS) distributes a group key to only authorized group members to encrypt group communications. The group key should be changed to prevent a joining/leaving member from accessing past/future communications. Since the group size may be very large, extensive research have been conducted to improve the scalability. Recently, Zhu [1] proposed a Synchro-Difference LKH (SD-LKH) scheme for scalable group key management, which is an improvement of the well-known LKH scheme [2]. This scheme optimizes the computational overhead of the KS, and the author claims that it achieves the same security level as the LKH scheme.

In this letter, we show that the SD-LKH scheme fails to provide the confidentiality of group communications as claimed, by presenting two kinds of active attacks. Specifically, in the first attack, the collusion of leaving adversaries can access future communications by discovering the next group key. Similarly, in the second attack, the collusion of joining adversaries can access past communications by discovering the previous group key. Based on these attacks, two or more unauthorized adversaries can collude and access communications within the group, which violates the purpose of secure multicast.

II. Review of the Synchro-Difference LKH Scheme

Zhu [1] presented the optimal key tree structure minimizing the computational overhead of the key server and proposed two kinds of schemes: the Iterated Hash Chain (IHC) scheme and the Synchro-Difference LKH (SD-LKH) scheme. Among them, the SD-LKH scheme has observably different strategy from other variations of the LKH scheme. The difference is that in rekeying process new keys are generated based on the previous ones by employing the distribution of the difference, instead of being randomly generated.

Initial key generation and distribution in the SD-LKH scheme are almost identical to those of the LKH scheme. First, the KS constructs a logical tree where each node represents a Key Encryption Key (KEK) that can be used to send rekeying messages securely to the members in the subtree rooted at the node. The key of the root node is appointed to a group key which is used to encrypt the communications within the group. Each group member corresponds to each leaf node and holds the set of keys along the path from the root to the leaf node. For example, a member u_1 in Fig. 1 holds $K_{3.1}$, $K_{2.1}$, $K_{1.1}$, and K_0, where K_0 is a group key and $K_{3.1}$ is his pair-wise key shared with the KS.

When a new member wants to join the group, the KS creates a new leaf node for him and shares the pair-wise key of the node with him. To prevent him from accessing past communications, *i.e.*, for backward security, the keys along the path from the root to the leaf node should be rekeyed. Suppose u_9 joins the group. After sharing $K_{3.9}$ with u_9, the KS randomly generates a differential value D and transmits the following rekeying messages:

$$\begin{array}{lcl} KS \rightarrow \{u_1, ..., u_8, u_{10}, ..., u_{27}\} & : & \{D\}_{K_0} \\ KS \rightarrow u_9 & : & \{K'_{2.3}, K'_{1.1}, K'_0\}_{K_{3.9}}. \end{array}$$

After receiving D, each member except for u_9 rekeys all and only the keys he is entitled to receive by respective XOR operations with D, *e.g.*, u_1 computes $K'_0 = K_0 \oplus D$, and $K'_{1.1} = K_{1.1} \oplus D$.

When an existing member leaves the group, the KS rekeys all the keys possessed by the member to prevent him from accessing future communications, *i.e.*, for forward security. Suppose u_9 leaves the group. The KS randomly generates D and distributes it to all members except for u_9 by transmitting the following messages:

$$\begin{array}{lcl} KS \rightarrow u_7 & : & \{D\}_{K_{3.7}} \\ KS \rightarrow u_8 & : & \{D\}_{K_{3.8}} \\ KS \rightarrow \{u_1, u_2, u_3\} & : & \{D\}_{K_{2.1}} \\ KS \rightarrow \{u_4, u_5, u_6\} & : & \{D\}_{K_{2.2}} \\ KS \rightarrow \{u_{10}, ..., u_{18}\} & : & \{D\}_{K_{1.2}} \\ KS \rightarrow \{u_{19}, ..., u_{27}\} & : & \{D\}_{K_{1.3}} \end{array}$$

After receiving D, each member except for u_9 rekeys all and only the keys he is entitled to receive in the same manner.

From the viewpoint of security, the author claims that its security level is equivalent to that of the LKH scheme, under the assumption that the generation of D is unpredictable by the members.

Manuscript received March 5, 2007. The associate editor coordinating the review of this letter and approving it for publication was Prof. Marc Fossorier. This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2005-042-D00294).

H. Kim, B. Chung, Y. Lee, and H. Yoon are with the CS Division, Korea Advanced Institute of Science and Technology, Korea (email: hykim@nslab.kaist.ac.kr).

Y. Park is with the College of Information and Communications at Hanyang University, Korea.

Digital Object Identifier 10.1109/LCOMM.2007.070325.

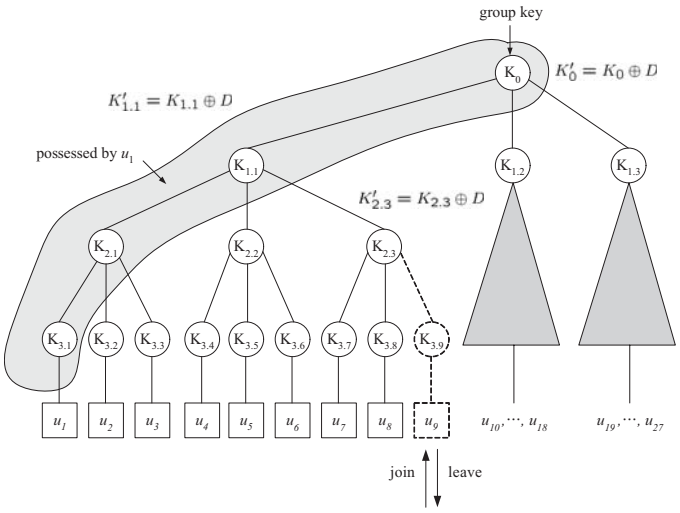

Fig. 1. Rekeying process in the SD-LKH scheme with $d = 3$ and $h = 3$.

III. Attacks on the Synchro-Difference LKH Scheme

The security flaw of the SD-LKH scheme is due to the fact that new keys are generated with the same difference, D. If a non-member having previous keys discovers the difference, he can also generate new keys from the previous ones. Based on this, we show that the SD-LKH scheme cannot provide forward and backward securiy by presenting two kinds of collusion attack scenarios. Since two or more adversaries can easily collude to discover group keys in many applications, the SD-LKH scheme should be modified to defend against these attacks.

A. Breaking Forward Security

The following scenario describes a kind of collusion attack in which two adversaries act as legitimate members during they are in the group and then cooperatively compute the next group key with very high probability after leaving the group. In Fig. 1, suppose that u_1 is the adversary A and u_8 is the adversary B. Let $\bar{K}$ be A's topmost KEK which is not entitled to B, *i.e.*, $K_{2.1}$ in this scenario. The detailed attack scenario is as follows.

1) A leaves the group keeping the KEKs K_0, $K_{1.1}$, $K_{2.1}$, $K_{3.1}$ which were previously transmitted from the KS.
2) B receives the following rekeying message from the KS:

$$KS \rightarrow \{u_7, u_8, u_9\} \quad : \quad \{D\}_{K_{2.3}}.$$

Then, B stores D and rekeys the following KEKs: $K'_0 = K_0 \oplus D$, $K'_{1.1} = K_{1.1} \oplus D$.

3) B leaves the group. A then overhears the following rekeying message which is transmitted from the KS through an open channel:

$$KS \rightarrow \{u_2, u_3\} \quad : \quad \{D'\}_{K'_{2.1}}$$

4) B informs D to A. Then, A computes rekeyed KEKs with D: $K'_0 = K_0 \oplus D$, $K'_{1.1} = K_{1.1} \oplus D$, $\bar{K}' = K'_{2.1} = K_{2.1} \oplus D$.
5) Now A can obtain D' by decrypting the rekeying message in step 3) with $\bar{K}'$. Then, he computes the next group key $K''_0 = K'_0 \oplus D'$, and thus can access future communications after B leaves. Moreover, A can hold the same next KEKs that u_2 holds as a legitimate member except for the pair-wise key: $K'_{2.1}, K''_{1.1} = K'_{1.1} \oplus D', K''_0$. Similarly, B can hold the next KEKs: $K'_{2.3} = K_{2.3} \oplus D', K''_{1.1}, K''_0$.

Let us consider how other members' join/leave operations affect the scenario. The fact that join operations do no harm to the scenario can be shown as follows. First, the join operations before the scenario starts do no harm obviously. Second, if new members join during the period between A's leave and B's leave, each differential value is distributed to current group members and B can always obtain it as a legitimate member. Then, B informs all differential values to A in step 4) and A can compute current $\bar{K}$ regardless of the changes in the tree structure. Third, if a new member joins after the attack succeeds, the adversaries can always obtain the differential value by decrypting the rekeying message with current group key they already know. Then, they can compute the next group key and rekey their KEKs with the differential value to prepare the next join/leave operation.

In case of leave operations, the scenario rarely fails to discover the next group key, which can be shown as follows. First, the leave operations before the scenario starts do no harm obviously. Second, if some members leave during the period between A's leave and B's leave, B receives all differential values as a legitimate member and A can also compute current $\bar{K}$ in step 4). However, if all members that $\bar{K}$ is entitled to (u_2 and u_3 in this scenario) leave the group, $\bar{K}$ will not be used by the KS in step 3) and thus this attack will fail. More formally, suppose a complete tree with degree d and height h. Assuming that the positions of A and B are randomly chosen ($A \neq B$), the probability that they share exactly l KEKs (including the root key), $P_{share}(l)$, is

$$P_{share}(l) = \frac{(d-1)d^{h-l}}{d^h - 1}.$$

Note that $\bar{K}$ is shared with A and other $d^{h-l} - 1$ members. In this case, when totally m members ($\neq A, B$) leave the group before B's leave, the probability that all these $d^{h-l} - 1$ members leave, $P_{leave}(l, m)$, is

$$P_{leave}(l,m) = \begin{cases} \frac{\binom{(d^h-2)-(d^{h-l}-1)}{m-(d^{h-l}-1)}}{\binom{d^h-2}{m}} & \text{if} \quad m \geq d^{h-l} - 1 \\ 0 & \text{if} \quad m < d^{h-l} - 1. \end{cases}$$

Thus, the average probability that this attack succeeds when m members leave the group during the period between A's leave and B's leave, $P_{succ}(m)$, is

$$P_{succ}(m) = 1 - \sum_{i=1}^{h} P_{share}(i) \cdot P_{leave}(i, m),$$

which is very high for large groups, *e.g.*, $P_{succ}(m) \geq 0.9$ for $m \leq 710$ in a group with $d = 3$, $h = 6$, and $d^h - 2 = 727$ members.

Third, suppose a member u leaves after the attack succeeds. Let $\bar{K}_A$ be A's topmost KEK which is not entitled to u, and

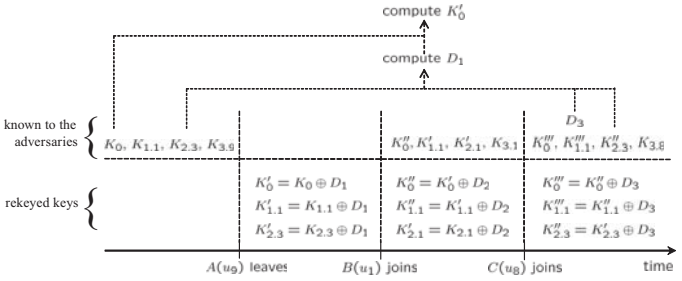

Fig. 2. Attack scenario to break backward security.

let $\bar{K_B}$ be B's topmost KEK which is not entitled to u. If either $\bar{K_A}$ or $\bar{K_B}$ is used by the KS for leave operation, the adversaries can obtain the differential value and thus compute the next group key and KEKs. For example, suppose that u_4 leaves the group just after B leaves. The KS transmits rekeying messages including the following:

$$KS \rightarrow \{u_2, u_3\} \quad : \quad \{D''\}_{K'_{2.1}}$$
$$KS \rightarrow \{u_7, u_9\} \quad : \quad \{D''\}_{K'_{2.3}}.$$

And the adversaries can obtain D'' with either A's $K'_{2.1}$ or B's $K'_{2.3}$. Then they are able to not only compute the next group key but also rekey their KEKs like other legitimate members such as u_2 or u_7:

$$K'''_0 = K''_0 \oplus D'', \quad K'''_{1.1} = K''_{1.1} \oplus D''.$$

If all members holding either $\bar{K_A}$ or $\bar{K_B}$ have left the group, neither of them is used by the KS to distribute the differential value and thus the adversaries fail to discover the next group key. However, since the probability that neither $\bar{K_A}$ nor $\bar{K_B}$ is used is also very low, the adversaries keep accessing communications with very high probability.

In summary, the adversaries discover the next group key with very high probability, and keep accessing communications with also very high probability even if other members join/leave the group after the attack finishes. The batch rekeying that manages multiple join/leave operations simultaneously is not dealt with in the SD-LKH scheme. However, we think that our attack still succeeds with acceptable probability though batch rekeying is applied.

The proposed attack does not succeed in the LKH scheme. After A leaves, the KS distributes randomly-generated KEKs (instead of D) to only proper members. For example in the scenario, the KEKs K_0, $K_{1.1}$, $K_{2.1}$ are rekeyed but B receives only K'_0, $K'_{1.1}$. Thus, A can not discover $\bar{K} = K'_{2.1}$ and then also can not obtain the next group key after B leaves. Therefore, the LKH scheme is secure against this attack unlike the SD-LKH scheme, which is contrary to the author's claim in [1].

B. Breaking Backward Security

We assume three adversaries A, B and C where initially A is participating in the group, and both B and C are outside the group. In the following attack scenario, the adversary A leaves the group, and then both B and C joins the group successively. The goal of them is to access the past communications during the period between A's leave and before B's join. In Fig. 1 suppose u_9 is the adversary A. The detailed attack scenario is as follows, and Fig. 2 describes the change of KEKs according to elapsed time.

1) A leaves the group keeping the KEKs $K_0, K_{1.1}, K_{2.3}, K_{3.9}$.
2) The KS distributes a differential value D_1, and the following keys are rekeyed:

   $$K'_0 = K_0 \oplus D_1, K'_{1.1} = K_{1.1} \oplus D_1, K'_{2.3} = K_{2.3} \oplus D_1.$$

   Then, A records the communications encrypted with K'_0.
3) B joins the group assuming that he is located at the position of u_1 whose leaf node is not in the subtree rooted at $K_{2.3}$. Before B's join, the KS distributes D_2 and the following keys are rekeyed:

   $$K''_0 = K'_0 \oplus D_2, K''_{1.1} = K'_{1.1} \oplus D_2, K'_{2.1} = K_{2.1} \oplus D_2.$$

   B receives K''_0, $K''_{1.1}$, $K'_{2.1}$, $K_{3.1}$ from the KS.
4) C joins the group assuming that he is located at the position of u_8 whose leaf node is in the subtree rooted at $K_{2.3}$. Before C's join, the KS distributes D_3 and the following keys are rekeyed:

   $$K'''_0 = K''_0 \oplus D_3, K'''_{1.1} = K''_{1.1} \oplus D_3, K''_{2.3} = K'_{2.3} \oplus D_3.$$

   B receives D_3 as a legitimate member. C receives K'''_0, $K'''_{1.1}$, $K''_{2.3}$, $K_{3.8}$ from the KS.
5) The adversaries collude and compute

   $$K''_{2.3} \oplus D_3 \oplus K_{2.3} = (K_{2.3} \oplus D_1 \oplus D_3) \oplus D_3 \oplus K_{2.3} = D_1.$$

   Finally, the group key K'_0 between A's leave and B's join can be computed with D_1: $K'_0 = K_0 \oplus D_1$. Thus, the adversaries can decrypt the recorded past communications with K'_0.

In step 3), we assume that A and B are located in different subtrees of the root node, where the probability is $\frac{d-1}{d}$. In step 4), we assume that A and C are located in the same subtree of the root node, where the probability is $\frac{1}{d}$. Therefore, the probability that this attack succeeds is $\frac{d-1}{d^2}$, which is an acceptable value.

Actually, this attack scenario is sensitive to other member's join/leave. If a legitimate member joins or leaves the group during the period between A's leave and C's join, it is possible that the attack fails to find out D_1 in step 5). However, it is valuable to consider this attack because the previous LKH scheme and other variations are not vulnerable to this kind of attack scenarios. Moreover, it is just one of many possible attack scenarios in which colluding adversaries discover KEKs not entitled to them, and it can be easily extended.

REFERENCES

[1] W. T. Zhu, "Optimizing the tree structure in secure multicast key management," *IEEE Commun. Lett.*, vol. 9, no. 5, pp. 477–479, 2005.
[2] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Networking*, vol. 8, no. 1, pp. 16–30, 2000.