Computer Networks 104 (2016) 16–26

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Towards collusion-attack-resilient group key management using one-way function tree

Yanming Sun[a], Min Chen[a,*], Abel Bacchus[b], Xiaodong Lin[b]

[a] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
[b] Faculty of Business and Information Technology, University of Ontario Institute of Technology, Oshawa, ON L1H 7K4, Canada

ARTICLE INFO

Article history:
Received 1 December 2014
Revised 23 September 2015
Accepted 16 April 2016
Available online 2 May 2016

Keywords:
Multicast communication
Group key management
Collusion attack
Vulnerability analysis
One-way function tree

ABSTRACT

One-way Function Tree (OFT) is a promising scheme for group key management. However, it has been found vulnerable to collusion attacks. Malicious users can collaborate to break forward and backward secrecy. Solutions have been proposed to prevent collusion attacks on OFT scheme. In this paper, we first demonstrate how existing solutions only partially consider collusion attacks. Current models surmise scenarios where malicious users may obtain node secrets unknown through collusion. They do not, however, consider that malicious users can decrypt extra blinded node secrets using known node secrets. As a result, the malicious users can collude to obtain far more information than expected. We use theoretical evidence to identify the exact node secrets which can be obtain by malicious users. Finally, we propose two improved schemes named repeated one-way function tree (ROFT) and node one-way function tree (NOFT). Compared to previous solutions, ROFT and NOFT require less adjustments to make the OFT scheme resilient to collusion attacks. Performance analysis shows that ROFT and NOFT do not incur extra communication overhead compared to the original OFT scheme. The proposed ROFT and NOFT schemes effectively solve the security problem of the OFT scheme at the cost of a minimal increase in computational cost and storage overhead.

1. Introduction

Multicast communication has been widely used in video conference technologies [1], interactive group games, and scientific discussions [2,3]. It is characterized by one sender to multiple receivers or multiple senders to multiple receivers. It can drastically improve message transmission efficiency. In multicast communication security services are provided to prevent malicious attackers from gaining unauthorized access to the messages [4]. Group key management is crucial for secure multicast; it provides key distribution, update, and storage. To guarantee the secure communication, group key management must preserve two security requirements: forward secrecy and backward secrecy [5]. For forward secrecy the group key should be updated as soon as a member leaves the group to prevent it from continuing to access the group's communication. In backward secrecy a new group key must be distributed to the group when a new member joins to prevent it from decoding previous content. At present, group key management is broadly classified into three categories. These include centralized group key management, distributed group key management, and de-centralized group key management [6–8].

One-way function tree (OFT) [9,10] is a typical centralized group key management scheme. It is based on logical key hierarchy (LKH) scheme [11,12] and takes key tree as key management structure. It adds one-way function to LKH. The communication overhead of the group manager when a member leaves is reduced to $\log_2 N$, where N is total number of users in the group. It is half of LKH.

However, Horng first showed in [13] that OFT is vulnerable to a type of collusion attack. Ku and Chen performed a study based on this attack and proposed other ways of collusion [14]. Xu *et al*. studied the possibility of collusion for two users [15]. Xu also derived the only node secrets that can be computed by a pair of colluding users and derived the preconditions for a successful collusion. In [16], Liu summarized previous researches and analyzed Xu's main propositions. Liu gave a counterexample to prove falsity of Xu's necessary and sufficient conditions for a collusion attack among a set of users on the OFT scheme.

In this paper, we studied the vulnerability of OFT based on the previous research. We found Xu's proposition, that node secrets obtained by a pair of colluding users, only partially considers the problem of collusion attacks. In his proposition, Xu ignored that the colluding users can decrypt extra blinded node secrets

* Corresponding author. Tel.: +86-13349999659.
E-mail addresses: yanming.epic@gmail.com (Y. Sun), minchen2012@hust.edu.cn (M. Chen), Abel.Bacchus@uoit.ca (A. Bacchus), xiaodong.lin@uoit.ca (X. Lin).

http://dx.doi.org/10.1016/j.comnet.2016.04.014
1389-1286/© 2016 Elsevier B.V. All rights reserved.

using node secrets obtained through collusion. We identify the exact node secrets which can obtained by colluding users. Then, we conclude Liu's counterexample cannot prove falsity of Xu's proposition. Finally, we propose two improved schemes named repeated one-way function tree (ROFT) and node one-way function tree (NOFT). Compared with other improved schemes, ROFT and NOFT require less adjustments to make the OFT scheme collusion attack resilient. The proposed ROFT and NOFT schemes effectively solve the security problem of the OFT scheme at the cost of a small increase in computational cost and storage overhead.

The remainder of the paper is organized as follows. Section 2 reviews related results. In Section 3, we analyze the problem in Xu's proposition concerning node secrets obtained by a pair of colluding users and identify the exact node secrets which can be obtained by colluding users. Based on this, we conclude Liu's proof against Xu's proposition as insufficient. In Section 4, we propose two improved schemes named ROFT and NOFT. Section 5 presents the security proof of ROFT and NOFT. Section 6 gives a comparison between our schemes and other related schemes. Section 7 summarizes this paper and gives the future work.

2. Related work

2.1. Related research

Horng showed in [13] that OFT is vulnerable to a type of collusion attack. A user leaving the group can obtain the blinded node secrets and use them for a collusion attack with a subsequently joining user. As a result, OFT cannot preserve forward secrecy and backward secrecy. Ku and Chen performed a study based on this premise and proposed other ways of collusion [14]. Furthermore, they developed an improved OFT scheme. In the scheme, when a user leaves the group, the blinded node secrets known by it are updated in addition to the update of node secrets. Their scheme can prevent collusion attacks, however, the cost is the group manager must broadcast h^2+h times when a user leaves, where h is the height of the key tree. Therefore, it creates additional communication overhead. To reduce the communication overhead of the group manager, Xu *et al.* studied the possibility of collusion for two users [15]. He found that any two users cannot always collude to obtain unknown key information. Therefore, every time a user leaves, the group manager does not always need to update all the blinded node secrets. Xu also derived the only node secrets that can be computed by a pair of colluding users and derived the preconditions for a successful collusion. On this basis, Xu proposed an improved scheme. In this scheme, the group manager records the leaving user and the blinded node secrets it knows. Every time a new user joins, the group manager decides whether it can collude with any user recorded. If possibility for collusion does not exist, the blinded node secrets need not to be updated. In this way, the communication overhead is reduced. However, in the scheme, the group manager needs to store large amounts of key information. The storage overhead of the group manager is linear to the size of the key tree. Although the communication overhead of Xu's scheme is lower than Ku's, it is still higher than the original OFT. In [16], Liu summarized previous researches and analyzed Xu's main propositions. Liu gave a counterexample to prove falsity of Xu's necessary and sufficient conditions for a collusion attack among a set of users on the OFT scheme. Then he put forward a new necessary and sufficient condition for nonexistence of any type of collusion attack. This attack is more catastrophic in healthcare or patient monitoring scenario [17,18], and environmental data collection architecture [26,27,29,30], especially for hot events detection [25,28]. In this respect, Cloud-assisted IoT-based secured framework was proposed for a healthcare system in [18], where the electrocardiogram (ECG) and other health-related data were obtained through IoT-enabled sensors, and watermarked, and then sent to the cloud. But we found the counterexample is not sufficient to prove the falsity of Xu's proposition.

Liu proposed an improved scheme of OFT named homomorphic one-way function tree (HOFT). HOFT uses the self-homomorphic function and modular multiplication in rekeying. It combines the key update when a user joins or leaves and the update for preventing collusion attacks. As a result, the key distribution for preventing collusion is simplified.

However, several issues have been found in HOFT. Node secrets in the tree are built over an Abelian group resulting in a distortion of the original OFT scheme. Using a trapdoor one-way function and modular multiplication instead of hash functions and exclusive-or operations creates a larger computational cost for HOFT. Therefore, key generation, management, and computation become relatively complex. In group key update, the group manager needs to perform several steps such as normalization, contraction, and expansion. These steps are complex to implement. For example, when a new user joins, the group manager needs to build a normalization tree and an expanding tree later combining them to form an incremental tree. To prevent collusion, a second increment tree is also needed. When a user leaves, the group manager needs to build a normalization tree and a contracting tree and then combines them to form an incremental tree. In [19], the author hoped to effectively prevent collusion attacks and reduce the computational overhead. He analyzed the results of previous studies and proposed a new scheme named secure group rekeying (SGRK) using exclusive-or operation and SHA-256. However, in SGRK, distributing the updated node secrets to the remaining users in the group was not considered. Therefore, the scheme has a large communication overhead and cannot preserve forward secrecy. Moreover, it cannot prevent collusion attacks. Such problem may become more serious when the network becomes complicate [20,21].

2.2. OFT scheme

OFT is a group key management scheme which was proposed by Sherman *et al.* in [9,10]. It is based on LKH scheme and uses one-way function in key management [22]. In OFT, there is a centralized group manager who is responsible for key updates, storage, and distribution. The management structure of OFT is a binary key tree. In this tree, each internal node i is associated with a node secret x_i, a blinded node secret y_i, and a node key K_i.

Definition 1. A key is called a blinded node secret which is computed by a one-way function on a node secret. In OFT, the blinded node secret is used for computing the upper node secrets in the key tree.

The node secret of the root is the group key. The blinded node secret y_i is computed by the formula $y_i = f(x_i)$ with x_i; the node key K_i is computed by the formula $K_i = g(x_i)$ with x_i, where f and g are different special one-way functions. K_i is used to encrypt the updated key information when rekeying. Every member in the group stores the blinded node secrets of the siblings of the nodes in the path from its leaf node to the root. Thus, every member can use its leaf node secret and the blinded node secrets to compute other node keys in the path in a bottom-up manner.

Referring to Fig. 1, i is an internal node in the key tree. Its left and right children are $2i$ and $2i+1$ respectively. The users in the subtree rooted at i can compute the node secret of i by the formula $x_i = y_{2i} \oplus y_{2i+1} = f(x_{2i}) \oplus f(x_{2i+1})$, where $\oplus$ is a bitwise exclusive-or operation. They can also compute the blinded node secret y_i by $y_i = f(x_i)$. These users store the blinded node secret y_s associated with node s which is the sibling of node i. Therefore, they can

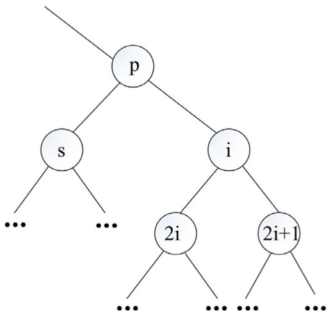

Fig. 1. Structure of OFT key tree.

compute the node secret of p by $x_p = y_s \oplus y_i$. By the same way, each member can compute the node secrets in the path from its node to the root, including the node secret associated with the root (the group key).

When a node secret in the key tree is changed, the new blinded node secret should be sent to the users who store the old blinded node secret in the group to complete the key update. For example, suppose that node secret x_i is changed. The new blinded node secret y_i should be sent to the users who store the old blinded secret of node i. These users are just the children of s. Since the children of s can compute the node key K_s, the group manager only needs to encrypt the new blinded node secret y_i with K_s. Then, the group manager should broadcast the encrypted key information to the group. Thus, the new blinded node secret will be sent to the appropriate group members.

When a new user joins the group, the group manager will choose a leaf node i which is nearest to the root. Node i will then be changed. The user associated with i becomes $2i$, the left child of i. The new user becomes $2i+1$, the right child of i. These two children will be distributed new node secrets. The new user will be sent the appropriate blinded node secrets. Finally, the node secrets from the position where the user joins to the root should be updated and the updated blinded node secrets should be sent to the appropriate users. The group manager need to send $2log_2N$ blinded node secret when a user joins.

When a user associated with i leaves, the node secrets from i to the root should be updated. If s, the sibling of i, is a leaf node, s will take the position of p which is the parent node of i. Then, the group manager will distribute a new node secret to s. If s is the root of a subtree, s will be moved to p to make the subtree rooted at s closer to the root. The node secret associated with a leaf node of the subtree will be changed to a new one. The updated blinded node secrets will be sent to the appropriate users in the group. The group manager need to send log_2N blinded node secret when a user leaves.

2.3. Collusion attack on the OFT scheme

In [13], Horng found that OFT is vulnerable to collusion attacks. Then, he concluded that OFT cannot preserve forward and backward secrecy.

Fig. 2 describes the changes in the membership in the group. Referring to Fig. 2, at first, Alice (associated with node 8) leaves at time t_A. Then, Bob joins at time t_B. Fig. 2 illustrates the key trees before Alice leaves, between t_A and t_B, and after Bob joins. There has no user leaves or joins between t_A and t_B. We use $x_{i[t_A,t_B]}$ to denote the node secret associated with node i in the time interval between t_A and t_B. We use $y_{i[t_A,t_B]}$ to denote the blinded node secret associated with node i in the time interval between t_A and t_B. After Alice leaves, x_3 is not changed until Bob joins. So, Alice holds its blinded version $y_{3[t_A,t_B]}$. x_2 is changed when Alice leaves and remains unchanged until Bob joins. When Bob joins, he receives its blinded version $y_{2[t_A,t_B]}$. Collectively, they know $y_{2[t_A,t_B]}$ and $y_{3[t_A,t_B]}$. Therefore, they can collude to obtain the group key in $[t_A, t_B]$ by computing $x_{1[t_A,t_B]} = y_{2[t_A,t_B]} \oplus y_{3[t_A,t_B]}$.

Alice can obtain the new group key after she leaves, so OFT fails to preserve forward secrecy. Bob can obtain the group key before he joins, so OFT fails to preserve backward secrecy.

According to this, Horng proposed two necessary conditions for a collusion attack to exist: 1) the two attackers must leave and join at different subtree of the root; 2) no key update happens in the time interval between t_A and t_B.

2.4. Researches on the collusion attack

Many scholars studied the collusion attack to OFT and drew different conclusions. In [14], Ku and Chen prove the two necessary conditions proposed by Horng are not really necessary. They gave out a new way for colluding by a pair of attackers.

To prevent the attacks, Ku and Chen proposed an improved scheme. When a user leaves, all the node secrets associated with the nodes in the path from the user to the root and the blinded node secrets of the siblings of the nodes in that path should be updated. Although the improved scheme can prevent collusion attack, it incurs a large communication overhead. When a user leaves, the group manager should send $(\log_2 N)^2 + \log_2 N + 1$ keys, whereas, in OFT, the group manager only needs to send $\log_2 N$ keys .

In [15], Xu found a pair of users cannot always collude to break OFT. The success of collusion depends on some relation between them.

Referring to Fig. 3, suppose A leaves at time t_A and C joins at time t_C after t_A. Let B, D, E, and F denote the subtrees rooted at nodes L, R, R', and R'' respectively. In the discussion below, let t_{DMIN}, t_{EMIN}, and t_{FMIN} denote the time of the first group key update happens in D, E, F after t_A. Let t_{BMAX}, t_{EMAX}, and t_{FMAX} denote the time of the last group key update happens in B, E, and F before t_C.

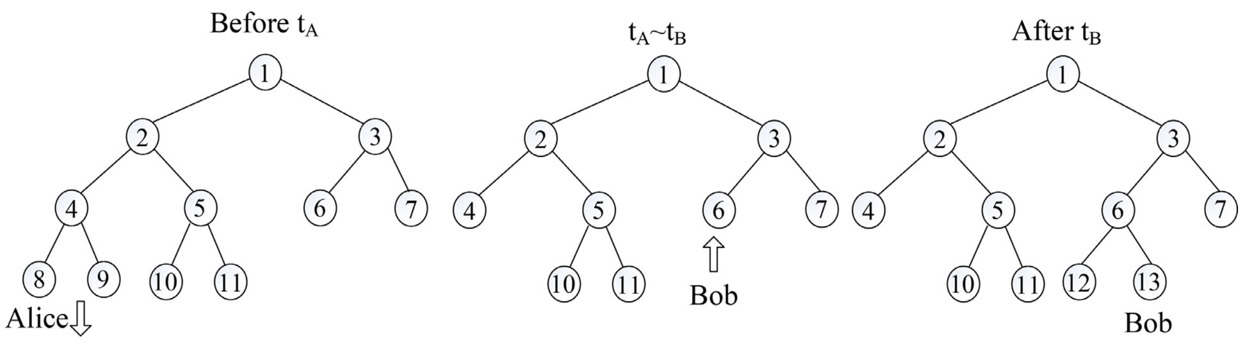

Fig. 2. Collusion attack on OFT.

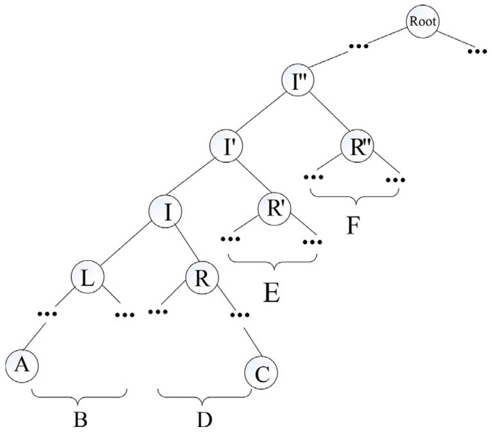

Fig. 3. Generic collusion attack on OFT.

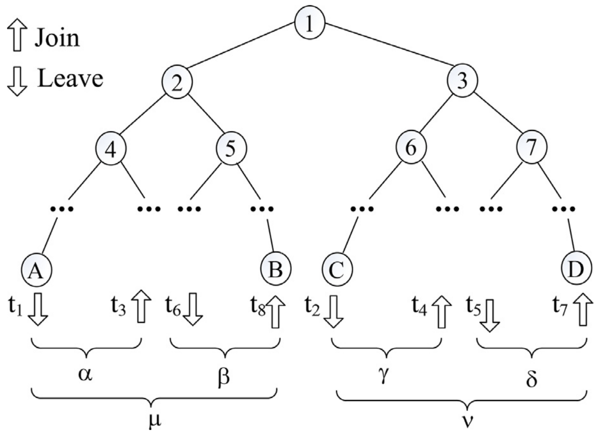

Fig. 4. Liu's counterexample against Xu's proposition 3.

Xu proposed the following propositions:

Xu's proposition 1: For the OFT scheme, referring to Fig. 3, the only node secrets that can be computed by A and C when colluding are:

$-x_I$ in the time interval $[t_{BMAX}, t_{DMIN}]$,

$-x_{I'}$ in $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMAX}, t_C])$,

$-x_{I''}$ in $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMAX}, t_C]) \cap ([t_A, t_{FMIN}] \cup [t_{FMAX}, t_C])$,

and so on, up to the root.

Xu's proposition 2: A pair of colluding users A and C cannot compute any node secret which they are not supposed to know by OFT, if one of the following conditions holds.

$-A$ is removed after C joins.

$-A$ and C both join.

$-A$ and C are both removed.

Xu's proposition 3: For OFT, an arbitrary collection of removed and joining users can collude to compute some unknown node secret, if and only if the same node secret can be computed by a pair of nodes in the collection.

In [16], Liu analyzed the propositions proposed by Xu and drew a different conclusion. Liu pointed out that Xu's proposition 3 is wrong. He claimed that the colluding users may not know those child blinded node secrets at first, but can collude to compute them. Therefore, in some specific conditions, some users in the collection can collude to obtain some node secrets that cannot be computed by any pair of users in the collection.

Liu gave a counterexample to prove falsity of Xu's proposition 3. Consider a collusion scenario depicted in Fig. 4. Suppose Alice (A) leaves at time t_1, Bob (B) joins at time t_8, Colin (C) leaves at time t_2, and Dean (D) joins at time t_7. The figure combines multiple key trees at different time t_i ($i = 1, \ldots, 8$). The chronological order of t_1, t_2, ..., and t_8 corresponds with the numerical order of their subscripts. Let α, β, γ, δ, μ, and ν denote the subtree rooted at node 4, 5, 6, 7, 2, and 3, respectively. In addition to the above changes, there are changes happened in α, γ, δ, and β at time t_3, t_4, t_5, and t_6. Let $t^X_{\alpha MAX}$ denote the time of the last group key update that happens in α before X joins. Let $t^Y_{\beta MIN}$ denote the time of the first group key update that happens in β after Y leaves. x_ν denotes the node secret associated with ν and y_ν denotes the blinded version of it. $x_{\nu[t_1,t_2]}$ denotes the node secret of ν in the interval $[t_1, t_2]$.

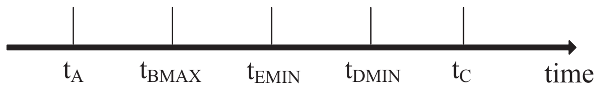

Fig. 5. Time order in OFT tree.

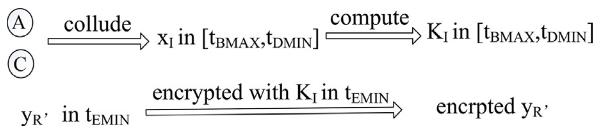

Fig. 6. Keys known by the attacker.

Liu's proof is as follows:

According to Xu's proposition 1, Alice and Bob can collude to compute x_2 in the time interval $[t^B_{\alpha MAX}, t^A_{\beta MIN}]$, i.e., $x_{2[t_3,t_6]}$; Colin and Dean can collude to compute x_3 in the time interval $[t^D_{\gamma MAX}, t^C_{\delta MIN}]$, i.e., $x_{3[t_4,t_5]}$. Thus, collectively, Alice, Bob, Colin, and Dean know $x_{2[t_3,t_6]}$ and $x_{3[t_4,t_5]}$. So, they can collude to compute $x_{1[t_4,t_5]}$.

According to Xu's proposition 1, all the node secrets that Alice and Bob can collude to compute are:

$-x_2$ in $[t^B_{\alpha MAX}, t^A_{\beta MIN}]$, i.e., $x_{2[t_3,t_6]}$,

$-x_1$ in $[t^B_{\alpha MAX}, t^A_{\beta MIN}] \cap ([t_1, t^A_{\nu MIN}] \cup [t^B_{\nu MAX}, t_8])$.

In the key tree of Fig. 4, $[t^B_{\alpha MAX}, t^A_{\beta MIN}] \cap ([t_1, t^A_{\nu MIN}] \cup [t^B_{\nu MAX}, t_8]) = [t_3, t_6] \cap ([t_1, t_2] \cup [t_7, t_8]) = \phi$. Thus, there is no time for Alice and Bob to compute x_1 by collusion. That is to say, Alice and Bob cannot collude to obtain $x_{1[t_4,t_5]}$. By the same argument, it can be proven that the collusion between Colin and Dean, that between Alice and Dean, or that between Colin and Bob, $x_{1[t_4,t_5]}$ cannot be computed. As a result, Liu concluded Xu's proposition 3 is wrong.

Collusion attack also occurs in participatory sensing or mobile crowd sensing, which can cause privacy leakage during data collection [24].

3. New condition for collusion attack

3.1. Analysis of current results

In Xu's proposition 1, he gave out all the node secrets that can be computed by a pair of colluding users A and C. But we found the proposition only partially consider collusion attacks. For the OFT scheme, referring to Figs. 3, 5, and 6, let t^{BMAX}_{EMIN} denote the time of the first group key update that happens in E after t_{BMAX}. Let t^{BMAX}_{FMIN} denote the time of the first group key update that happens in F after t_{BMAX}.

In Xu's proposition 1, if a group key update happens in E, according to OFT, the group manager must update the node secret $x_{R'}$. Then, the new blinded version $y_{R'}$ must be encrypted with K_I, the node key associated with I at that time, and the group manager broadcasts it. Since A and C can collude to obtain x_I in $[t_{BMAX}, t_{DMIN}]$, they can compute K_I in $[t_{BMAX}, t_{DMIN}]$. Thus, they can decrypt the key update information to obtain $y_{R'}$. They can then obtain $x_{I'}$ by computing $x_{I'} = y_I \oplus y_{R'}$.

3.2. Analysis of Collusion Attack

According to the discussion in the previous section, we draw the following conclusions:

Theorem 1. *Referring to Fig. 3, the only node secrets that can be obtained by A and C when colluding are:*

−x_I in $[t_{BMAX}, t_{DMIN}]$,

−$x_{I'}$ in $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMIN}^{BMAX}, t_{DMIN}] \cup [t_{EMAX}, t_C])$,

−$x_{I''}$ in $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMIN}^{BMAX}, t_{DMIN}] \cup [t_{EMAX}, t_C]) \cap ([t_A, t_{FMIN}] \cup [t_{FMIN}^{BMAX}, t_{DMIN}] \cup [t_{FMAX}, t_C])$,

and so on, up to the root.

Proof. To prove the theorem, two cases will be discussed. □

Case 1: In $[t_A, t_{BMAX}]$, group key update has not occurred in E. From t_A to t_{BMAX}, $x_{R'}$ has not been updated. So, A knows the blinded node secret associated with R', $y_{R'}$ at t_{BMAX}. Then, in $[t_{BMAX}, t_{DMIN}]$, if $x_{R'}$ has not been updated yet, A knows $y_{R'}$. Otherwise, if $x_{R'}$ has been updated, every time $x_{R'}$ is updated, the blinded version $y_{R'}$ will be encrypted with K_I at that time, and the group manager broadcasts it. According to Xu's proposition 1, A and C can collude to get x_I in $[t_{BMAX}, t_{DMIN}]$. So, A can obtain K_I in $[t_{BMAX}, t_{DMIN}]$ and all the updated $y_{R'}$ in the time interval of $[t_{BMAX}, t_{DMIN}]$. That is to say, no matter the blinded node secret associated with R' has been updated or not, A can still get $y_{R'}$ in $[t_{BMAX}, t_{DMIN}]$. Therefore, A and C can compute $x_{I'}$ in $[t_{BMAX}, t_{DMIN}]$ by $x_{I'} = y_I \oplus y_{R'}$.

Case 2: In $[t_A, t_{BMAX}]$, group key update has happened in E. From t_A to t_{BMAX}, $x_{R'}$ has been updated. So, A does not know $y_{R'}$ at t_{BMAX}. At this time, there are three cases. Case a: $t_{EMAX} < t_{BMAX}$. Recall t_{EMAX} denote the time of the last group key update that happens in E after t_A. So, in $[t_{EMAX}, t_C]$, there has not been a group key update in E. That is to say, $x_{R'}$ has not been updated in $[t_{EMAX}, t_C]$. Therefore, C can get $y_{R'}$ in $[t_{EMAX}, t_C]$. Since $t_{EMAX} < t_{BMAX}$ and $t_{DMIN} < t_C$, $[t_{BMAX}, t_{DMIN}] \subset [t_{EMAX}, t_C]$. C can get $y_{R'}$ in $[t_{BMAX}, t_{DMIN}]$. According to Xu's proposition 1, A and C can collude to obtain x_I in $[t_{BMAX}, t_{DMIN}]$. They can compute y_I in $[t_{BMAX}, t_{DMIN}]$. So, C can compute $x_{I'}$ in $[t_{BMAX}, t_{DMIN}]$ by $x_{I'} = y_I \oplus y_{R'}$. Case b: $t_{EMAX} > t_{BMAX}$ and there has happened group key update in E in $[t_{BMAX}, t_{DMIN}]$ at least once. That is to say, $x_{R'}$ has been updated in $[t_{BMAX}, t_{DMIN}]$. After t_{EMIN}^{BMAX}, the first time $x_{R'}$ is updated, A can obtain $y_{R'}$ in $[t_{EMIN}^{BMAX}, t_{DMIN}]$ by decrypting the rekeying information with K_I. Since A and C can collude to obtain y_I in $[t_{BMAX}, t_{DMIN}]$, A can compute $x_{I'}$ in $[t_{EMIN}^{BMAX}, t_{DMIN}]$. Since A can obtain $y_{R'}$ in $[t_A, t_{EMIN}]$ and C can obtain $y_{R'}$ in $[t_{EMAX}, t_C]$, they can compute $x_{I'}$ in $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMAX}, t_C])$. Therefore, they can compute $x_{I'}$ in $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMIN}^{BMAX}, t_{DMIN}] \cup [t_{EMAX}, t_C])$. Case c: $t_{EMAX} > t_{BMAX}$ and there has not happened a group key update in E in $[t_{BMAX}, t_{DMIN}]$. That is to say, $x_{R'}$ has not been updated in $[t_{BMAX}, t_{DMIN}]$, whereas in $[t_{DMIN}, t_C]$, $x_{R'}$ has been updated. Now, A and C cannot obtain $y_{R'}$ in $[t_{BMAX}, t_{DMIN}]$. So, they cannot obtain $x_{I'}$.

To sum up, after a simple merge, we conclude A and C can collude to obtain $x_{I'}$ in $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMIN}^{BMAX}, t_{DMIN}] \cup [t_{EMAX}, t_C])$, where t_{EMIN}^{BMAX} may equal to t_{EMIN}. The $x_{I'}$ which A and C can collude to obtain is more than Xu's proposition 1.

Similarly, A and C can collude to obtain $x_{I''}$ in $[t_{BMAX}, t_{DMIN}] \cap ([t_A, t_{EMIN}] \cup [t_{EMIN}^{BMAX}, t_{DMIN}] \cup [t_{EMAX}, t_C]) \cap ([t_A, t_{FMIN}] \cup [t_{FMIN}^{BMAX}, t_{DMIN}] \cup [t_{FMAX}, t_C])$.

Based on the proof of Theorem 1, we examine Liu's counterexample in Fig. 4 as follows.

Recall that Liu concluded that Alice and Bob can collude to compute $x_{2[t_3,t_6]}$; Colin and Dean can collude to compute $x_{3[t_4,t_5]}$. But any pair of them cannot obtain $x_{1[t_4,t_5]}$ by collusion.

At t_4, a user joins in γ. x_3 is updated. The group manager encrypts y_3 with K_2 and then broadcasts it. Since $t_3 < t_4 < t_6$ and Alice can collude with Bob to compute $x_{2[t_3,t_6]}$, they can compute K_2 at t_4 with $x_{2[t_3,t_6]}$. Thus, Alice and Bob can decrypt the rekeying information to obtain $y_{3[t_4,t_5]}$. Then, they can compute $x_{1[t_4,t_5]}$ by $x_{1[t_4,t_5]} = y_{2[t_4,t_5]} \oplus y_{3[t_4,t_5]}$. This contradict Liu's proof. As a result, we conclude that Liu's counterexample cannot prove the falsity of Xu's proposition 3.

Theorem 2. *Referring to Fig. 3, if a pair of colluding users A and C can obtain any node secret which they are not supposed to know by the OFT scheme, the following conditions must be true.*

−C joins after A leaves or A joins after C leaves.

−There are enough users in B and D.

Suppose C joins after A leaves. "There are enough users in B and D" means from A's leaving to C's joining, B will not disappear due to user's leaving and D should exist before C joins.

Proof. We can get the first condition from Xu's proposition 2. So, only the necessary of the second condition is proved as follows. □

When there is only a member A in the group, the theorem is correct. If after A leaves, B disappears due to user's leaving. According to OFT, when the last users in B leaves, D will be moved closer to the root. Node I will be substituted by R and R becomes the left child of I'. Then, the blinded node secret associated with R will be updated. Thus, C cannot obtain y_L in $[t_{BMAX}, t_C]$. As a result, A and C cannot obtain any unknown node secrets by collusion.

If before C joins, D does not exist. That is to say, when C joins, there is no user in B and D, and a user is in the position of I in the tree. According to OFT, when C joins, C and the user associated with I will be allocated to the position of L and R. Now, there is only a user C in D. Thus, A cannot obtain node secrets not already known by collusion with C.

4. Improvement on the OFT scheme

In OFT scheme, when a new user joins, it will be sent the blinded node secrets which were once used to compute the past group key. On the other hand, when a user leaves, it brings out the blinded node secrets that may be used to compute the future group key. Thus, a leaving and joining user pair can combine their knowledge to compute a valid group key between the time of leaving and that of joining. Therefore, to prevent collusion attacks, two solutions can be used. They are to prevent a leaving user from bringing out any blinded node secrets that contain information about future group key and to supply a joining user with the blinded node secrets that contain no information about the past group key. These solutions break the necessary conditions of collusion attacks. Base on the second solution, we propose two improved schemes named ROFT and NOFT.

4.1. ROFT

The process of group key management is as follows. Suppose there are 6 users in the group. The key tree structure is as Fig. 7.

When a user Bob joins, the group manager chooses a leaf node which is nearest to the root for him. In Fig. 7, the node is 6. Then, the group manager allocates Bob to 13 and the original user associated with 6 will be allocated to 12. The group manager sends Bob a new node secret and updates the node secret associated with 12. Subsequently, the group manager notifies the users in the group that a new user joins. In this section, the path from 13 to the root is called Bob's key path.

To prevent collusion attacks, when Bob joins, the group manager should update the blinded node secrets associated with the siblings of the nodes in Bob's key path. The users who know the

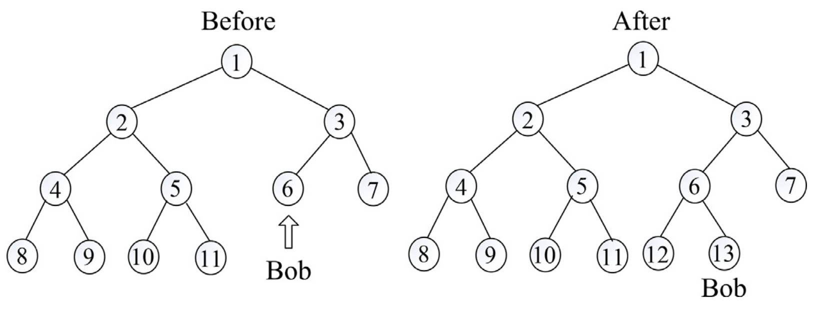

Fig. 7. ROFT key tree.

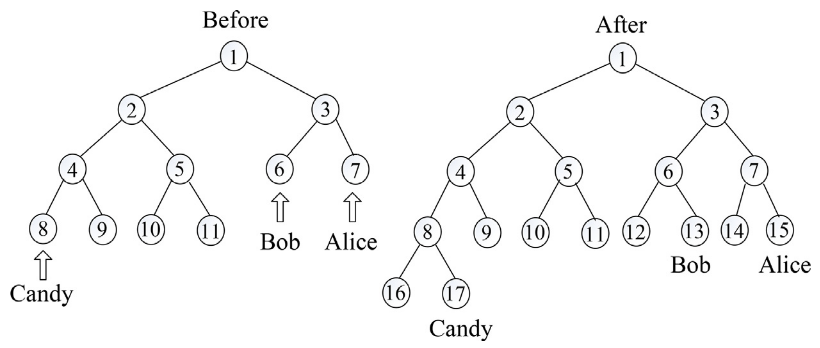

Fig. 8. Adding multiple members in ROFT.

blinded node secrets should do a one-way function on them and store the results as the new blinded node secret. In the key update after that, if the blinded node secret is stored, it is used for computing directly. For example, suppose before Bob joins, the node secret associated with 2 is x_2 and the blinded version of it is y_2. Recall $y_2 = f(x_2)$. The node secret associated with 7 is x_7 and the blinded version of it is y_7. Recall $y_7 = f(x_7)$. Users associated with 7, 8, 9, 10, 11, and 12 know y_2. When they receive the message that Bob joins, they store $f(f(x_2))$ as the blinded node secret associated with 2. User associated with 7 and 12 knows y_7, so they stores $f(f(x_7))$ as the blinded node secret associated with 7. Then, the group manager sends the blinded node secret associated with 12, $f(x_{12})$, the blinded node secret associated with 7, $f(f(x_7))$, and the blinded node secret associated with 2, $f(f(x_2))$ to Bob. Now, Bob can only know these blinded node secrets. He cannot obtain $f(x_2)$ from $f(f(x_2))$ and cannot obtain $f(x_7)$ from $f(f(x_7))$, too.

When a user who is in the subtree rooted at a node leaves or a user joins under the subtree rooted at a node, all the users in the subtree should delete the blinded node secret associated with the node if it is stored. For example, suppose that after Bob joins, a user joins at the position of 8. The blinded node secret associated with 2 should be updated. Now, the users associated with 8, 9, 10, and 11 can delete the stored blinded node secret associated with 2. Users associated with 7, 12, and 13 can obtain the new blinded node secret associated with 2 by decrypting the key update messages.

Batch group rekeying requires operation that can process multiple membership changes at the same time. To describe adding multiple members simultaneously, taking the scenario depicted in Fig. 8 as an example. To add member Bob, Alice, and Candy to the group, the group manager needs to perform the following steps. 1) The group manager adds the new member to the key tree. 2) The group manager changes all the blinded node secrets before supplying them to the new members.

To supply the new member the changed blinded node secrets, the siblings of the nodes in the key path of Bob, Alice, and Candy should change their blinded node secrets. However, the node secrets in the key path of the new members are changed to new, the blinded node secrets in the path are new too. So, only the siblings of the nodes in the key path of new member which hold old blinded node secrets need to change their blinded node secrets. Therefore, in Fig. 8, only node 5 and node 9 needs to change their blinded node secrets. In the key tree, the users who know the old blinded node secret of node 5 should do a one-way function on it and store the results as the new blinded node secret. In Fig. 8, user 9, 10, 11, and 17 need to compute the new blinded node secret of 5 which is $f(f(x_5))$ as the new blinded node secret of node 5 and store it. Similarly, the users who know the old blinded node secret of node 9 should do a one-way function on it and store the results as the new blinded node secret. In Fig. 8, user 9 and 17 need to store $f(f(x_9))$ as the new blinded node secret of node 9.

To distribute the new node secrets and blinded node secrets to the old member, the group manager needs to encrypt new x_{12}, x_{14}, and x_{17} with the old node key K_6, K_7, and K_8 respectively and send them to the users. In addition, the group manager needs to send $\{f(x_{16})\}\_K_{17}$, $\{f(x_8)\}\_K_9$, $\{f(x_4)\}\_K_5$, $\{f(x_2)\}\_K_3$, $\{f(x_{13})\}\_K_{12}$, $\{f(x_6)\}\_K_7$, $\{f(x_3)\}\_K_2$, $\{f(x_{15})\}\_K_{14}$, $\{f(x_7)\}\_K_6$ to the group. The group manager also needs to supply every new member with blinded node secrets it is entitled to, i.e., $\{f(x_{12}), f(x_7), f(x_2)\}\_K_{13}$, $\{f(x_{14}), f(x_6), f(x_2)\}\_K_{15}$, $\{f(x_{17}), f(f(x_9)), f(f(x_5)), f(x_3)\}\_K_{16}$.

To describe removing multiple users simultaneously, taking the scenario depicted in Fig. 9 as an example. The members need to be removed from the group are Bob, Alice, and Candy. To remove Bob, Alice, and Candy, the group manager needs to delete the nodes of them and move the siblings of them to the parent nodes.

To distribute the new node secrets and the blinded node secrets to the old member, the group manager needs to encrypt new x_4, x_6, and x_7 with the old node key K_9, K_{12}, and K_{14} respectively and send them to the users. In addition, the group manager needs to send $\{f(x_4)\}\_K_5$, $\{f(x_2)\}\_K_3$, $\{f(x_6)\}\_K_7$, $\{f(x7)\}\_K_6$, $\{f(x_3)\}\_K_2$ to the group.

When a user who is in the subtree rooted at a node leaves, all the users in the subtree should delete the blinded node secret associated with the node if it is stored.

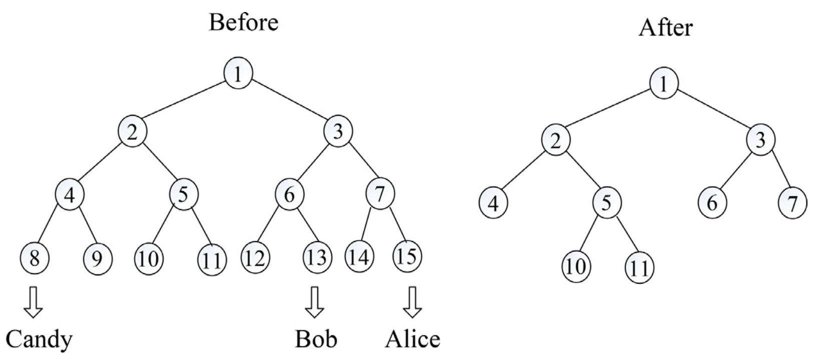

Fig. 9. Removing multiple members in ROFT.

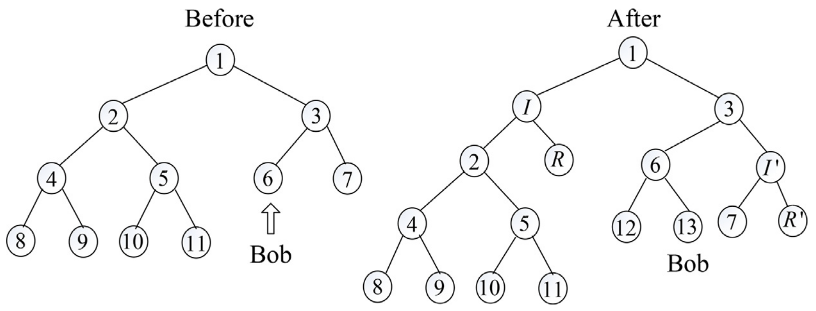

Fig. 10. NOFT key tree.

4.2. NOFT

The process of group key management is as follows. Suppose there are 6 users in the group. The key tree structure is as Fig. 10.

To prevent collusion attacks, when Bob joins, the group manager should change the blinded node secrets associated with the siblings of the nodes in Bob's key path. Referring to Fig. 10, the group manager adds a new node I at the original position of node 2. Now, node 2 becomes the left child of I. The right child of I is a virtual node R. The initial node secret of R, x_R, is set to 1. Here, all the users who know the blinded node secret of 2 store the node secret of R. In Fig. 10, the users are 7, 8, 9, 10, 11, and 12. They can compute the node secret associated with I by $x_I = f(x_2) \oplus f(x_R)$ and compute its blinded version by $y_I = f(x_I)$. Similarly, the group manager adds a new node I' at the original position of node 7. Then the group manager sends $f(x_{12})$, $f(x_I)$, and $f(x_{I'})$ to Bob.

After Bob joins, if a new user joins under the subtree rooted at node 3, to prevent the key tree from growing too big, the group manager will do a one-way function on x_R. That is to say, when a new user joins and it is allocated to the subtree rooted at 3, $f(x_R)$ should be the new node secret of R. Then, the users who know the blinded node secret associated with R need to update it.

When a new user joins, the group manager can choose R as the position and R can be substituted by the new user. Then the node secret and the blinded node secret will be updated.

In NOFT, the group manager should distinguish the internal node and virtual node. When a group key update happens in the subtree rooted at 3, the new blinded node secret associated with 3, y_3, should not be encrypted using K_I, but using K_2.

When the node secret associated with the sibling of R is changed, R and I can be deleted. For example, referring to Fig. 10, after Bob joins, a user joins and is allocated under the subtree rooted at 2, the node secret associated with 2 should be updated. Now, R and I can be deleted. Then, node 2 substitutes the position of I and becomes the child of node 1. The updated blinded node secret associated with 2 should be sent to the appropriate users.

To describe adding multiple members simultaneously, taking the scenario depicted in Fig. 11 as an example. To add member Bob, Alice, and Candy to the group, the group manager needs to perform the following steps. 1) The group manager adds the new member to the key tree. 2) The group manager adds new nodes I to the key tree.

To prevent collusion attack of the users, the siblings of the nodes in the key path of Bob, Alice, and Candy should change their blinded node secrets. However, the node secrets in the key path of the new members are changed to new, the blinded node secrets in the path are new too. So, only the siblings of the nodes in the key path of new member which hold old blinded node secrets need to add new node I. Therefore, in Fig. 8, the group manager adds a new node I at the original position of node 5. Now, node 5 becomes the left child of I. The right child of I is a virtual node R. All the users who know the blinded node secret of 5 store the node secret of R. In Fig. 11, the users are 9, 10, 11, and 17. Similarly, the group manager adds a new node I' at the original position of node 9. Node 9 becomes the left child of I'. The right child of I' is a virtual node R'. All the users who know the blinded node secret of 9 store the node secret of R'. In Fig. 11, the users are 9 and 17. The initial node secrets of R and R' are set to 1.

To distribute the new node secrets and the blinded node secrets to the old member, the group manager needs to encrypt new x_{12}, x_{14}, and x_{17} with the old node key K_6, K_7, and K_8 respectively and send them to the users. In addition, the group manager needs to send $\{f(x_{16})\}\_K_{17}$, $\{f(x_8)\}\_K_9$, $\{f(x_4)\}\_K_5$, $\{f(x_2)\}\_K_3$, $\{f(x_{13})\}\_K_{12}$, $\{f(x_6)\}\_K_7$, $\{f(x_3)\}\_K_2$, $\{f(x_{15})\}\_K_{14}$, $\{f(x_7)\}\_K_6$ to the group. The group manager also needs to supply every new member with blinded node secrets it is entitled to, i.e., $\{f(x_{12}), f(x_7), f(x_2)\}\_K_{13}$, $\{f(x_{14}), f(x_6), f(x_2)\}\_K_{15}$, $\{f(x_{17}), f(I'), f(I), f(x_3)\}\_K_{16}$.

The procession of removing multiple users simultaneously in NOFT is similar with ROFT. So, it is omitted. The difference is when the node secret associated with the sibling of R is changed, R and I can be deleted.

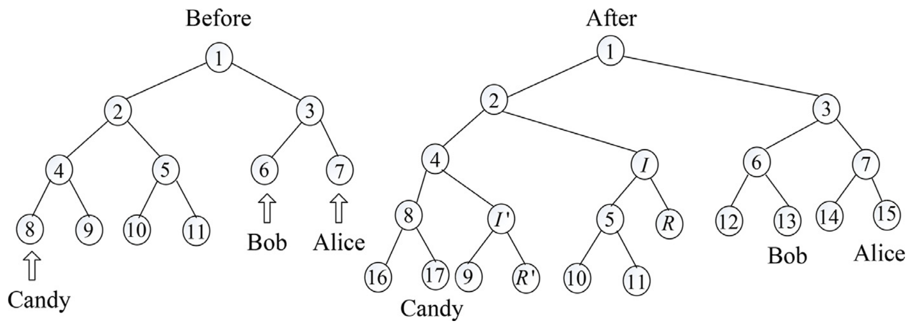

Fig. 11. Adding multiple members in NOFT.

5. Security proof

Panjwani develops a symbolic security model in [23] for studying group key management protocols. We prove the security of ROFT under this model. Consider a group of n users, labeled 1,2,…,n. Each user i shares a private key K_i with the group manager. At time t, users in a set $S^{(t)} \subseteq \{1, 2, \dots, n\}$ refer to the users in the group at that time. The ROFT tree corresponding to $S^{(t)}$ is denoted by $Tr^{(t)}$. Let $[n]$ denote the set $\{1, \dots, n\}$ and $2^{[n]}$ denote the power set of $[n]$. The group dynamic up to time t can be represented by a sequence of sets $\overrightarrow{S^{(t)}} = (S^{(0)}, S^{(1)}, \dots, S^{(t)}) \in (2^{[n]})^t$. If for all $t \geq 1$, $S^{(t-1)}$ changes into $S^{(t)}$ through a single change, the sequence $\overrightarrow{S^{(t)}}$ is called simple. Refer to [16], we presents the following definition.

Definition 2. An n-user OFT-based protocol is called correct, if for all $t \geq 0$, for simple sequence $\overrightarrow{S^{(t)}}$, $\forall i \in S^{(t)}$, i only knows the node secrets on the path from its associated leaf node to the root and the blinded node secrets that are siblings to this path, and no other node secrets nor blinded node secrets in $Tr^{(t)}$.

Definition 3. An n-user OFT-based protocol is called secure against single user attack, if for all $t \geq 0$, for simple sequence $\overrightarrow{S^{(t)}}$, $\forall i \notin S^{(t)}$, i can never recover any node secret in $Tr^{(t)}$ from K_i and the rekeying messages.

Definition 4. An n-user OFT-based protocol is called secure against collusion attacks, if for all $t \geq 0$, for simple sequence $\overrightarrow{S^{(t)}}$, an arbitrary set of user $Col = \{i | i \notin S^{(t)}\}$, Col can never recover any node secret in $Tr^{(t)}$ from $\{K_i | i \in Col\}$ and the rekeying messages.

Theorem 3. *ROFT protocol is correct and secure against single user attack.*

Proof. We use induction to prove this claim over t. For $t = 0$, since $S^{(0)} = \phi$, the claim is true. Now we discuss that if the claim is true for $t - 1 \geq 0$ then it is true for t as well. For simple sequence $\overrightarrow{S^{(t)}} = (S^{(0)}, S^{(1)}, \dots, S^{(t)}) \in (2^{[n]})^t$, we only consider the following cases:

Case (1) $(i \in S^{(t-1)} \wedge i \in S^{(t)})$: In ROFT, i can only recover those rekeyed node secrets and blinded node secrets it holds in $Tr^{(t-1)}$ as required from rekeying message. Therefore, it holds and only holds all the node secrets and blinded node secrets in $Tr^{(t)}$ as required.

Case (2) $(i \notin S^{(t-1)} \wedge i \in S^{(t)})$: That is to say, i joins at time t. In ROFT, new joining member i can only recover the required node secrets and blinded node secrets from the rekeying messages.

Case (3) $(i \in S^{(t-1)} \wedge i \notin S^{(t)})$: That is to say, i is removed at time t. From hypothesis, i only knows the node secrets on its path to the root and the blinded node secrets of the sibling to this path in $Tr^{(t-1)}$. In ROFT, all the node secrets in i's path to the roots in $Tr^{(t-1)}$ are changed at time t. Although some blinded node secrets in $Tr^{(t-1)}$ that i brings out may not change at time t, the rekeying messages is encrypted with the node secrets of the siblings to i's path to the root in $Tr^{(t-1)}$. Therefore, i cannot know any node secret on $Tr^{(t)}$.

Case (4) $(i \notin S^{(t-1)} \wedge i \notin S^{(t)})$: That is to say, i is removed before time t-1. From the hypothesis, i can never know any node secret in $Tr^{(t-1)}$. Since the rekeying messages in t is encrypted by a node secret in $Tr^{(t-1)}$, i cannot recover the rekeying messages. Thus, i can never compute any node secret on $Tr^{(t)}$. □

Theorem 4. *An arbitrary pair colluding users A and C cannot compute any node secret unknown by ROFT.*

Proof. We first consider the case C joins after A is removed. Referring to Fig. 3, suppose A is removed at time t_A and later C joins at time t_C. According to ROFT, all the blinded node secrets of the siblings to C's path to the root are changed at t_C. Therefore, C cannot know any blinded node secrets which is used before it joins the group. Thus, A and C cannot compute any node secret unknown. Second, we consider the case A and C join at the same time. According to ROFT, all the node secrets on the key path of A and C are changed. A and C cannot compute any node secret unknown with the blinded node secrets they know. The case in which A and C are removed at the same time is similar with this. It is omitted. In summary, A and C cannot compute any node secret unknown by ROFT. □

Liu's Theorem 1: For a one-way function tree X with n members, an arbitrary collection of $k(2 \leq k \leq n)$ parties cannot collude to compute any node secret unknown (including the group key), if and only if an arbitrary pair of parties cannot collude to compute any node secret unknown.

This Theorem is given and proved by Liu in [16].

Theorem 5. *ROFT protocol is secure against collusion attack.*

Proof. We prove this claim by contradiction. Suppose there exists a $t_0 \geq 0$ and a set of attackers which are not in $S^{(t_0)}$ can obtain a node secret x_i in $Tr^{(t_0)}$. Since ROFT is secure against single user attack, each attacker can never obtain x_i in $Tr^{(t_0)}$. The attackers must collude to obtain x_i. According to Liu's Theorem 1, there must exist a pair of attackers who can collude to compute a node secret unknown. This contradicts Theorem 4. Therefore, ROFT protocol is secure against collusion attack. □

The proof that NOFT is secure against collusion attack is similar with ROFT. It is omitted for the limited of space.

Table 1
Adding a member.

AC	OFT No	Ku&Chen Yes	Xu Yes	HOFT Yes	ROFT Yes	NOFT Yes
OG	$(2\log_2 N+1)*L$	$(2\log_2 N+1)*L$	$(2\log_2 N+1)*L$ or $((\log_2 N)^2+2\log_2 N+1)*L$	$(2\log_2 N+1)*L$	$(2\log_2 N+1)*L$	$(2\log_2 N+1)*L$
OC	$(2\log_2 N+1)*C_E+(\log_2 N+1)*C_h$	$(2\log_2 N+1)*C_E+(\log_2 N+1)*C_h$	$(2\log_2 N+1)*C_E+(\log_2 N+1)*C_h$ or $((\log_2 N)^2+2\log_2 N+1)*C_E+((\log_2 N)^2+\log_2 N+1)*C_h$	$(2\log_2 N+1)*C_E+(1+\log_2 N+S(\log_2 N))*C_f+2S(\log_2 N)+1)*C_M$	$(2\log_2 N+1)*C_E+2\log_2 N*C_h$	$(2\log_2 N+1)*C_E+2\log_2 N*C_h$
OU	$2C_D+\log_2 N*C_h$	$2C_D+\log_2 N*C_h$	$2C_D+\log_2 N*C_h$ or $(\log 2N+1)*C_D+0.5*(\log_2 N)^2*C_h$	$(1+\log_2 N)*C_D+(\log_2 N+S(\log_2 N))*C_f+2\log_2 N*C_M$	$2C_D+(2\log_2 N-1)*C_h$	$2C_D+(2\log_2 N-1)*C_h$

Table 2
Removing a member.

AC	OFT No	Ku&Chen Yes	Xu Yes	HOFT Yes	ROFT Yes	NOFT Yes
OG	$(\log_2 N+1)*L$	$((\log_2 N)^2+\log_2 N+1)*L$	$(\log_2 N+1)*L$	$(\log_2 N+1)*L$	$(\log_2 N+1)*L$	$(\log_2 N+1)*L$
OC	$(\log_2 N+1)*C_E+\log_2 N*C_h$	$((\log_2 N)^2+\log_2 N+1)*C_E+((\log_2 N)^2+\log_2 N)*C_h$	$(\log_2 N+1)*C_E+\log_2 N*C_h$	$(\log_2 N+1)*C_E+(\log_2 N-1)*C_f+(\log_2 N+2)*C_M$	$(\log_2 N+1)*C_E+\log_2 N*C_h$	$(\log_2 N+1)*C_E+\log_2 N*C_h$
OU	$C_D+\log_2 N*C_h$	$\log_2 N*C_D+0.5*(\log_2 N)^2*C_h$	$C_D+\log_2 N*C_h$	$C_D+\log_2 N*C_f+(\log_2 N+1)*C_M$	$C_D+\log_2 N*C_h$	$C_D+\log_2 N*C_h$

Table 3
Storage overhead.

	OFT	Ku&Chen	Xu	HOFT	ROFT	NOFT
SU	$(2\log_2 N+1)*L$	$(2\log_2 N+1)*L$	$(2\log_2 N+1)*L$	$(2\log_2 N+1)*L$	$3\log_2 N*L$	$(2\log_2 N+2)*L$

6. Comparison with other schemes

There are several schemes to improve OFT. We compare the efficiency of the schemes from communication overhead, computational cost, and storage overhead.

According to [9,14–16], we give a comparison between our schemes and related schemes, covering: preventing collusion attack (AC), group manager's communication overhead(OG), group manager's computational cost (OC), maximum member computational cost (OU), and maximum member storage overhead (SU).

In Tables 1, 2, and 3, Ku&Chen is referred to the improved scheme proposed by Ku *et al.* Xu is referred to the scheme proposed by Xu *et al.* HOFT is the scheme proposed by Liu. L is the size of a cryptographic key or (blinded) node secret in bits. N is the number of total members in the group. $S(l)$ is the size of the combined ancestor tree (CAT) induced by l leaving (joining or changing) members, as in [10]. C_E, C_D, C_h, C_f, and C_M denote the computational cost of one encryption function, one decryption function, one hash function, one trapdoor one-way function, and one modular multiplication, respectively.

In ROFT, the group manager's communication overhead is the same as that in OFT. This is mainly because no extra key information needs to be sent by the group manager in rekeying. When a new member joins, to prevent collusion attack, the users in the group only require one-way functions. Compared to OFT, in ROFT the group manager incurs an additional computational cost for computing $\log_2 N-1$ blinded node secrets. Members in the group incur extra storage overhead for storing $\log_2 N-1$ blinded node secrets at most. Since $C_h < C_f$ and $\log_2 N < S(\log_2 N)$, the group manager's computational cost and the members' maximum computational cost in ROFT are lower than that in HOFT.

In NOFT, the group manager's communication overhead is the same as that in OFT. This is due to the default value of the virtual node. No extra key information needs to be sent. Since a virtual node is added, the height of the key tree increases by 1. The storage overhead of the members in the group is one more than that of OFT. More importantly, since $C_h < C_f$ and $\log_2 N < S(\log_2 N)$, the group manager's computational cost and the members' maximum computational cost in NOFT are lower than that in HOFT.

By analysis, we conclude that ROFT and NOFT proposed in this paper only incur small increase in the group manager's computational cost and the storage overhead of members, compared to the OFT scheme. The communication overhead of the group manager is the same as OFT. More importantly, the communication overhead and computational cost of ROFT and NOFT are lower than that of other schemes.

7. Conclusion

In this paper, we studied the security vulnerabilities of the OFT scheme. Firstly, we summarized the main conclusions on this subject and analyzed the conflicts found in other conclusions. Based on this analysis, we identified the exact node secrets which malicious users can obtain by collusion. Additionally, we provided proof for our own conclusion. Secondly, we proposed two improved OFT schemes named ROFT and NOFT. Compared with OFT, ROFT and NOFT do not incur extra communication overhead on the group manager. They can make the OFT scheme collusion attack resilient. More importantly, in ROFT and NOFT, the group manager has lower communication overhead and computational cost than in other schemes.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (no. 61572220).

References

[1] G. Fortino, W. Russo, C. Mastroianni, C.E. Palau, M. Esteve, CDN-supported collaborative media streaming control, IEEE MultiMedia 14 (2) (2007) 60–71.
[2] J. Chen, K. He, R. Du, Y. Xiang, Dominating set and network coding-based routing in wireless mesh networks, IEEE Trans. Parallel Distrib. Syst. 26 (2) (2016) 423–433.
[3] X. Ge, S. Tu, G. Mao, C.X. Wang, T. Han, 5g ultra-dense cellular networks, IEEE Wireless Commun. 23 (1) (Feb. 2016) 72–79.
[4] J. Chen, K. He, R. Du, F. Yu, Q. Yuan, L. Wang, A multi-objective optimization model based on immune algorithm in wireless mesh networks, Int. J. Commun. Syst. 29 (1) (2016) 155–169.
[5] B. Jiang, X. Hu, A survey of group key management, in: IEEE International Conference Computer Science and Software Engineering, Wuhan, China, 2008, pp. 994–1002.
[6] S. Rafaeli, D. Hutchison, A survey of key management for secure group communication, ACM Comput. Surv. 35 (3) (2003) 309–329.
[7] L. Zhou, D. Wu, B. Zheng, M. Guizani, Joint physical-application layer security for wireless multimedia delivery, IEEE Commun. Mag. 52 (3) (2014) 66–72.
[8] L. Zhou, H.C. Chao, A. Vasilakos, Joint forensics-scheduling strategy for delay-sensitive multimedia applications over heterogeneous networks, IEEE J. Sel. Areas Commun. 29 (7) (2011) 1358–1367.
[9] A.T. Sherman, D.A. McGrew, Key establishment in large dynamic groups using one-way function trees, IEEE Trans. Softw. Eng 29 (5) (2003) 444–458.
[10] D. Balenson, D. McGrew, A. Sherman, Key Management For Large Dynamic Groups: One-Way Function Trees and Amortized Initialization, Draft-irtf-S-mug-Groupkeymgmt-Oft-00.Txt, Internet Research Task Force, 2000.
[11] D.M. Wallner, E.J. Harder, R.C. Agee, Key Management for Multicast: Issues and Architectures, Internet Draft, Internet Engineering Task Force, 1998.
[12] C.K. Wong, M. Gouda, S.S. Lam, Secure group communication using key graphs, IEEE/ACM Trans. Netw. 8 (1) (2000) 16–30.
[13] G. Horng, Cryptanalysis of a key management scheme for secure multicast communications, IEICE Trans. Commun E85-B (5) (2002) 1050–1051.
[14] W.C. Ku, S.M. Chen, An improved key management scheme for large dynamic groups using one-way function trees, in: Proceedings International Conference Parallel Processing Workshops, Kaohsiung, Taiwan, 2003, pp. 391–396.
[15] X. Xu, L. Wang, A. Youssef, B. Zhu, Preventing collusion attacks on the one-way function tree (OFT) scheme, in: Proceedings 5th International Conference Applied Cryptography and Network Security, Zhuhai, China, 2007, pp. 177–193.
[16] J. Liu, B. Yang, Collusion-resistant multicast key distribution based on homomorphic one-way function trees, IEEE Trans. Inf. Forensics Security 6 (3) (2011) 980–991.
[17] M.S. Hossain, Cloud-supported cyber-physical localization framework for patients monitoring, IEEE Syst. J. (2016).
[18] M.S. Hossain, G. Muhammad, Cloud-assisted industrial internet of things (IIot)-enabled framework for health monitoring, Comput. Netw. (2016), doi:10.1016/j.comnet.2016.01.009.
[19] S. Abbas, P. Aravind, Collusion-free multicast group key distribution based on tree structures, Int. J. Adv. Innovative Res. 2 (11) (2013) 92–100.
[20] K. Zheng, L. Hou, H. Meng, Q. Zheng, N. Lu, L. Lei, Soft-defined heterogeneous vehicular network: architecture and challenges, IEEE Network (to be published in July, 2016), arXiv preprint arXiv:1510.06579.
[21] L. Lei, Z. Zhong, K. Zheng, J. Chen, H. Meng, Challenges on wireless heterogeneous networks for mobile cloud computing, IEEE Wireless Commun. 20 (3) (2013) 34–44.
[22] K. He, J. Chen, R. Du, Q. Wu, G. Xue, X. Zhang, Deypos: deduplicatable dynamic proof of storage for multi-user environments, IEEE Trans. Comput. (2016).
[23] S. Panjwani, Private group communication: two persepectives and a unifying solution, Computer Science Engineering Department, University of California. San Diego, CA, 2007.
[24] B. Zhang, C.H. Liu, J. Lu, Z. Song, Z. Ren, J. Ma, W. Wang, Privacy-preserving qoi-aware participant coordination for mobile crowdsourcing, Comput. Netw. (2016).
[25] C.H. Liu, J. Zhao, H. Zhang, S. Guo, K.K. Leung, J. Crowcroft, Energy-efficient event detection by participatory sensing under budget constraints, Syst. J. PP (99) (2016) 1–12.
[26] C.H. Liu, B. Zhang, X. Su, J. Ma, W. Wang, K.K. Leung, Energy-aware participant selection for smartphone-enabled mobile crowd sensing, Syst. J. PP (99) (2015) 1–12.
[27] C.H. Liu, J. Fan, H. Pan, J. Wu, K.K. Leung, Toward qoi and energy efficiency in participatory crowdsourcing, IEEE Trans. Veh. Technol. 64 (10) (2015) 4684–4700.
[28] B. Zhang, Z. Song, C.H. Liu, J. Ma, W. Wang, An event-driven qoi-aware participatory sensing framework with energy and budget constraints, ACM Trans. Intell. Syst. Technol. 6 (3) (2015) 42.
[29] C.H. Liu, J. Fan, J. Branch, K.K. Leung, Toward qoi and energy-efficiency in internet-of-things sensory environments, IEEE Trans. Emerg. Topics Comput. 2 (4) (2014) 473–487.
[30] C.H. Liu, B. Yang, T. Liu, Efficient naming, addressing and profile services in internet-of-things sensory environments, Ad Hoc Netw. 18 (2014) 85–101.

Yanming Sun received the M.S. degree in computer application from Northeastern University, Shenyang, China, in 2005 and the Ph.D. degree from Institute of Software, Chinese Academy of Sciences, Beijing, China, in 2013, respectively. Since December 2013, he has been with Huazhong University of Science and Technology, Wuhan, China, as a Postdoctoral Fellow. His current research interests include network security, vehicular ad hoc network security, privacy for wireless networks, etc.

Min Chen is a professor in School of Computer Science and Technology at Huazhong University of Science and Technology (HUST). He is Chair of IEEE Computer Society (CS) Special Technical Communities (STC) on Big Data. He was an assistant professor in School of Computer Science and Engineering at Seoul National University (SNU) from Sep. 2009 to Feb. 2012. He worked as a Post-Doctoral Fellow in Department of Electrical and Computer Engineering at University of British Columbia (UBC) for three years. Before joining UBC, he was a Post-Doctoral Fellow at SNU for one and half years. He received Best Paper Award from IEEE ICC 2012, and Best Paper Runner-up Award from QShine 2008. He serves as editor or associate editor for Information Sciences, Wireless Communications and Mobile Computing, IET Communications, IET Networks, Wiley I. J. of Security and Communication Networks, Journal of Internet Technology, KSII Trans. Internet and Information Systems, International Journal of Sensor Networks. He is managing editor for IJAACS and IJART. He is a Guest Editor for IEEE Network, IEEE Wireless Communications Magazine, etc. He is Co-Chair of IEEE ICC 2012-Communications Theory Symposium, and Co-Chair of IEEE ICC 2013-Wireless Networks Symposium. He is General Co-Chair for the 12th IEEE International Conference on Computer and Information Technology (IEEE CIT-2012) and Mobimedia 2015. He is General Vice Chair foTridentcom 2014. He is Keynote Speaker for CyberC 2012, Mobiquitous 2012 and Cloudcomp 2015. He has more than 260 paper publications, including 120+ SCI papers, 50+ IEEE Trans./Journal papers, 6 ISI highly cited papers and 1 hot paper. He has published two books: OPNET IoT Simulation (2015) and Big Data Inspiration (2015) with HUST Presss, and a book on big data: Big Data Related Technologies (2014) with Springer Series in Computer Science. His Google Scholars Citations reached 6,050+ with an h-index of 38. His top paper was cited 720+ times, while his top book was cited 420 times as of Aug. 2015. He is an IEEE Senior Member since 2009. His research focuses on Cyber Physical Systems, IoT Sensing, 5G Networks, Mobile Cloud Computing, SDN, Healthcare Big Data, Medica Cloud Privacy and Security, Body Area Networks, Emotion Communications and Robotics, etc.

Abel Bacchus received a BITs honours degree from the University of Ontario Institute of Technology (UOIT), Oshawa, Ontario, Canada in 2014. He is currently working towards his master of science degree in Computer Science at UOIT. His current research interests include cryptosystems, intuitive consent schemes and privacy enhancement mechanisms for E-health, and hardening network security for the internet of things.

Xiaodong Lin received the PhD degree in information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1998, and the PhD degree (with Outstanding Achievement in Graduate Studies Award) in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2008. He is currently an Associate Professor of information security with the Faculty of Business and Information Technology, University of Ontario Institute of Technology, Oshawa, ON, Canada. His research interests include wireless network security, applied cryptography, computer forensics, software security, and wireless networking and mobile computing. He is a member of the IEEE.