FDLKH: Fully Decentralized Key Management Scheme on Logical Key Hierarchy

Daisuke Inoue and Masahiro Kuroda

National Institute of Information and Communications Technology,
3-4, Hikarino-oka, Yokosuka, Kanagawa, 239-0847, Japan
{dai, marsh}@nict.go.jp

Abstract. In the next generation mobile environment, which uses 4G cellular and high-speed wireless LANs, novel group communication services among mobile users are expected to grow up. Security technologies for these group type services are challenging research area, especially, decentralization of group key management is important for large users. In this paper, we propose a fully decentralized key management scheme FDLKH that provides a key updating mechanism for dynamic group without any central server. This scheme inherits the key updating mechanism of the Logical Key Hierarchy scheme LKH, which is based on a central server system, and extends the LKH not to expect any central server but to use representative members of a group called captains. On the FDLKH, the total variety of keys in a group is half of that of the LKH. The costs for a member join or leave keep the logarithmic order of the number of members.

1 Introduction

The Internet and wireless communications, such as 3G cellular and wireless LAN, have infiltrated into daily life and next generation mobile environment using 4G cellular and high-speed wireless LANs is expected to come up in the market. In current mobile environment, there are many client-server type secure services, such as mobile e-commerce[1]. These services maintain a central server for secure communications with many users and expect high bandwidth communications for serving users. The next generation mobile environment[2], on the other hand, is said to be flat rated wireless access taking advantage of each wireless communication and novel group communication services among mobile users are expected to grow up in near future. In these group type services, mobile users exchange information securely in a group, where the number of mobile devices is large and many devices have less computational power comparing to computer systems. These services expect dynamic and secure communications because of member changes in a group. Security technologies to satisfy requirements for these services are challenging research area. In particular, decentralization of security management is important for large users. A mechanism for updating a group key, which is shared among all members in a large and dynamic group, is an important issue.

M. Jakobsson, M. Yung, J. Zhou (Eds.): ACNS 2004, LNCS 3089, pp. 339–354, 2004.

The group key should be updated when there is a join or leave in a group not to allow the new member (newcomer hereafter) to access past information and not to allow the left member (seceder hereafter) to access future information in the group, in other words, to provide backward and forward secrecy. There are two types of strategies. One is to update keys in a central system and to redistribute from the system[3]-[11]. The other is to agree a key among all members of a group by means of extended Diffie-Hellman (DH hereafter) key agreement protocols[12]-[16].

The former centralized key management, which targets for large and dynamic groups, concentrates tasks for updating the group key in a central server. There are some issues, such as maintenance cost, service level resilience and key exposure. Since the central server is required high computational resources and bandwidth, the server maintenance costs high. When there is a failure at the central server, services will not be available. When the server is compromised and keys are wrongly distributed to non group members, all the information among the group will be exposed to public. Furthermore, the centralized key management system asks all members to trust the central server; this may allow inspection of the system manager.

The latter extended DH based key management, which aims at relatively small groups, does not need a central server, but imposes computationally intensive modular exponentiations on all members for each join/leave procedure. These expensive computations may not be feasible for a group consisting of mobile devices with less computational power.

In this paper, we propose a fully decentralized key management scheme (FDLKH hereafter) that provides a group key update mechanism for group communications. Our scheme has two design principles: 1) No central server; 2) Some mobile devices having high computation power capable of modular exponentiations. The FDLKH inherits the group key update mechanism of the Logical Key Hierarchy scheme (LKH hereafter), which has been developed as a key management mechanism based on a central server system, and extends the LKH not to expect any central server. The FDLKH proposes to use representative members in a group, called *captains*, instead of a central server, with less network traffics. Only *captains* perform the DH key agreements and each *captain* distributes keys to adequate members. In this scheme, the costs of the DH key agreements and symmetric key encryptions/decryptions keep the logarithmic order of the number of members.

This paper is organized as follows. In Section 2, we describe the centralized key management schemes. We explain the FDLKH scheme in Section 3, and evaluate the scheme in Section 4. Lastly in Section 5, we provide our conclusion and future works.

2 Centralized Key Management Scheme

The centralized key management scheme maintains a group key, a symmetric key, shared among all members of a secure communication group at a server.

When a new member joins or some member leaves the group, the key is updated to provide the backward and forward secrecy. Here we explain two centralized group key management schemes.

2.1 Basic Scheme

The basic scheme to update a group key is that a central server and each group member share an individual key, which is used to encrypt/decrypt the group key. When a newcomer joins a group, the server generates a new group key and encrypts it by two keys, the old group key and an individual key of the newcomer, separately. The server sends the encrypted new group key by using the old group key to the existing members and to the newcomer by using the individual key. When a member leaves the group, the server generates a new group key and encrypts it by using the individual keys of all members except the seceder and sends every encrypted group key to each member. It is obvious that this scheme is not scalable, especially when a member leaves, because the computation at the server for the group key encryption is proportional to the number of remaining members.

2.2 Logical Key Hierarchy Scheme

The Logical Key Hierarchy scheme[5][6] provides a scalable mechanism to update a group key with a central server. The LKH employs a hierarchical tree structure that places each member of a group at each leaf. An intermediate node of the tree is associated with a key that is used to encrypt another key, a key encryption key. A leaf node is also associated with a key, which is an individual key of a member. The key associated with the root node of the tree is the group key. A member has all keys associated with all nodes from the parent node to the root node, called ancestor nodes. Essentially, the LKH divides members into subtrees that are rooted at each intermediate node and contain all descendants of its root node. Members of a subtree share a key associated with the top nodes of the subtree. When a newcomer joins or a seceder leaves a group, the keys associated with the ancestor nodes of the newcomer/seceder are updated. In case of a join, the central server encrypts new keys by using old keys of the ancestor nodes and distributes the encrypted new keys to corresponding subtrees. In case of a leave, the server encrypts new keys by using keys associated with the siblings of the ancestor nodes; especially for sibling members of the seceder, the server uses their individual keys to encrypt the new keys. Thereby encryptions are lumped together for each subtree, as a result, cryptographic rekeying computations becomes $\mathrm{O}(\log n)$ instead of $\mathrm{O}(n)$, where n is the number of members. Although the LKH provides a way to update the group key, a central server is responsible for the key generation and distribution.

3 FDLKH Scheme

In this section, we describe our scheme FDLKH: fully decentralized key management scheme on a logical key hierarchy. The FDLKH inherits the key hierarchy of the LKH and extends the key management not to expect any central server. The LKH employs a tree with degree d as a hierarchical tree, whereas the FDLKH uses a binary key tree that logically maps symmetric keys into intermediate nodes in the tree and places each member at each leaf. Dissimilarly to members on the LKH, members on the FDLKH have no individual keys, because the FDLKH expects no keys to share with a central server. To allot some portion of tasks of a central server, the FDLKH proposes a *captain* who is one of members in a group that represents a subtree in a binary key tree. When there is a join or a leave in a group, a *captain* is selected for each subtree. Then the *captains* perform key agreements and key distributions. They use the DH key agreement protocol for the key agreements and uses the LKH approach as mentioned in section 2.2 for the key distributions. In other words, the FDLKH combines the DH key agreement with the LKH to establish a rekeying mechanism without a central server.

3.1 Notations

$\langle l, m\rangle$	m-th node at level l in a tree, where $0 \le m \le 2^l - 1$
$\mathrm{M}_{\langle l,m\rangle}$	Member who occupies the node $\langle l, m\rangle$
$\mathrm{K}_{\langle l,m\rangle}$	Key associated with the node $\langle l, m\rangle$
$\mathrm{K}'_{\langle l,m\rangle}$	New key to be associated with the node $\langle l, m\rangle$
$\mathrm{T}_{\langle l,m\rangle}$	Subtree rooted at the node $\langle l, m\rangle$
$\mathrm{C}_{\langle l,m\rangle}$	Member who represents the $\mathrm{T}_{\langle l,m\rangle}$ (i.e., *captain*)
E(K, X)	Encryption of data X using a symmetric key K
A $\leftrightarrow$ B: K	A and B agree a symmetric key K by the DH key agreement
A $\rightarrow$ B: X	A sends data X to B by using multicast or unicast(s)
n	Number of members in a group
i, j, k	Integers $i = 1, \ldots, l, \quad j = 2, \ldots, l, \quad k = 2, \ldots, l-1$
p	Prime number
g	Generator of Z_p^*

In a binary key tree for the FDLKH, a node is identified by its level l ($l = 0, 1, 2, \ldots$) that the node belongs to and by the position m ($0 \le m \le 2^l - 1$) that is numbered from the leftmost grid in that level. Here a node is described as $\langle l, m\rangle$. The node $\langle 0, 0\rangle$ is the root of the tree. Members in a group occupy the leaf nodes of the tree. We denote a member who occupies the node $\langle l, m\rangle$ as $\mathrm{M}_{\langle l,m\rangle}$. Every intermediate node in the tree, a node unoccupied by a member, is associated with a symmetric key. A key mapped into the node $\langle l, m\rangle$ is $\mathrm{K}_{\langle l,m\rangle}$. The key is shared among all members who belong to the subtree rooted at the node $\langle l, m\rangle$. The subtree is described as $\mathrm{T}_{\langle l,m\rangle}$. The key $\mathrm{K}_{\langle 0,0\rangle}$, the group key, is shared among all members.

In Fig. 1(a), two members, $\mathrm{M}_{\langle 5,8\rangle}$ and $\mathrm{M}_{\langle 5,9\rangle}$, belong to the subtree $\mathrm{T}_{\langle 4,4\rangle}$ and share the symmetric key $\mathrm{K}_{\langle 4,4\rangle}$. They also belong to subtrees $\mathrm{T}_{\langle 3,2\rangle}$, $\mathrm{T}_{\langle 2,1\rangle}$,

$T_{\langle 1,0\rangle}$, and $T_{\langle 0,0\rangle}$. In total, the members have five keys: $K_{\langle 4,4\rangle}$, $K_{\langle 3,2\rangle}$, $K_{\langle 2,1\rangle}$, $K_{\langle 1,0\rangle}$, and $K_{\langle 0,0\rangle}$. A member knows all keys associated with its ancestor nodes up to the root node.

When there is a join or a leave in a group, some members are selected as representatives of subtrees to perform key agreements and key distributions. We call the representative member as a *captain*. The *captain* of the subtree $T_{\langle l,m\rangle}$ is denoted as $C_{\langle l,m\rangle}$.

Fig. 1(b) is a generalized binary key tree of the FDLKH, which is illustrated from the viewpoint of the member $M_{\langle l,m\rangle}$. The parent node of $M_{\langle l,m\rangle}$ is described as $\langle l-1, \lfloor \frac{m}{2} \rfloor \rangle$. The sibling member of $M_{\langle l,m\rangle}$ is $M_{\langle l,m+(-1)^m\rangle}$. All ancestor nodes of $M_{\langle l,m\rangle}$ are generalized as $\langle l-i, \lfloor \frac{m}{2^i} \rfloor \rangle$, where $i = 1, \ldots, l$. The subtrees that $M_{\langle l,m\rangle}$ belongs to and the keys that $M_{\langle l,m\rangle}$ has are, therefore, generalized as $T_{\langle l-i, \lfloor \frac{m}{2^i} \rfloor \rangle}$ and $K_{\langle l-i, \lfloor \frac{m}{2^i} \rfloor \rangle}$.

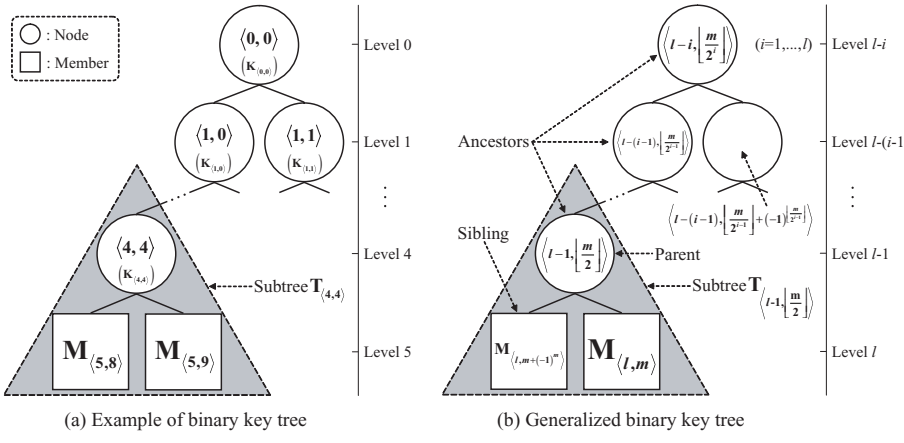

Fig. 1. Binary key tree

3.2 Flow of Join and Leave Protocol

The operations of the join and leave protocol are as follows:

***Join*:** 1) Tree update, 2) Captain selection, 3) DH key agreement, 4) Key distribution, 5) Key update

***Leave*:** 1) Captain selection, 2) DH key agreement, 3) Key distribution, 4) Key update, 5) Tree update

3.3 Tree Update

When a newcomer joins or a seceder leaves a group, each member independently updates the binary key tree of itself. At the first step of a join, the information

of a binary key tree that expresses positions of all members in the tree is sent to the newcomer by a *captain* who represents the subtree rooted at the parent node of the newcomer. The reorganizations of a key tree resulting from a join and a leave of a member are shown in Fig. 2 and Fig. 3.

A newcomer joins in a tree at the shallowest and leftmost leaf. In the case of Fig. 2, a newcomer becomes the member $M_{\langle 3,3\rangle}$. Owing to this join, $M_{\langle 2,1\rangle}$ who once occupied the node $\langle 2,1\rangle$ moves to the deeper level in the tree, thereby becomes the left child of the node $\langle 2,1\rangle$, namely $M_{\langle 3,2\rangle}$. In the special case, when the tree has a pyramidal shape, that is, the number of members before a newcomer join is some power of 2, the tree forms a complete binary tree, the root node $\langle 0,0\rangle$ is changed to $\langle 1,0\rangle$, the subtree $T_{\langle 0,0\rangle}$ is moved to $T_{\langle 1,0\rangle}$, and a new root node is created. After this reorganization, the newcomer becomes the right child of $\langle 0,0\rangle$, namely $M_{\langle 1,1\rangle}$.

The join point of a newcomer in a tree is decided unambiguously, however, a seceder leaves a tree at an arbitrary point. In the case of Fig. 3, the member $M_{\langle 3,0\rangle}$ leaves the tree. The member $M_{\langle 3,1\rangle}$, who is the sibling of $M_{\langle 3,0\rangle}$, moves to a shallower level in the tree and becomes $M_{\langle 2,0\rangle}$. Incidentally, in case where a sibling node of a seceder is unoccupied by a member, the subtree rooted at the sibling node moves to a lower level bodily and updates the name of its nodes and also the name of associated keys. For example, if the member $M_{\langle 2,3\rangle}$ in Fig. 3 leaves the tree, the subtree $T_{\langle 2,2\rangle}$ moves up and becomes $T_{\langle 1,1\rangle}$.

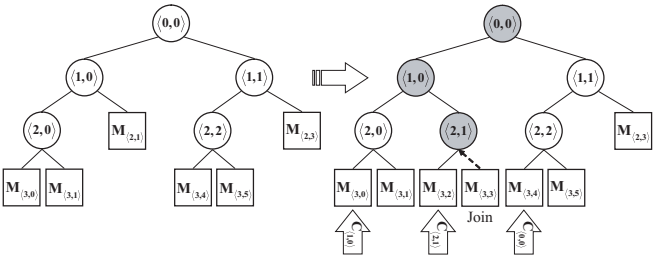

Fig. 2. Member join

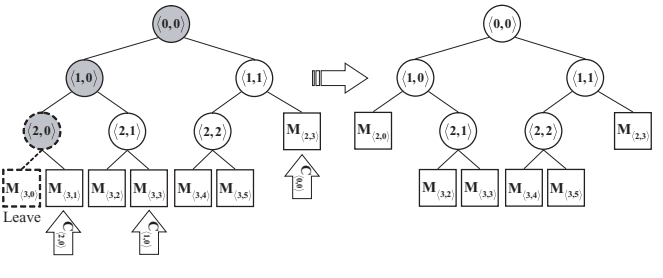

Fig. 3. Member leave

3.4 Rekeying Node

Rekeying nodes, namely nodes having keys to be updated, are all ancestor nodes of a newcomer or those of a seceder. The key generation, the key alternation, and the key destruction are performed on the keys of the rekeying nodes to provide the backward and forward secrecy.

In Fig. 2, the nodes $\langle 2,1\rangle$, $\langle 1,0\rangle$, and $\langle 0,0\rangle$ are the rekeying nodes, so that the key associated with $\langle 2,1\rangle$ is newly generated and other keys are altered to new ones. In Fig. 3, the nodes $\langle 2,0\rangle$, $\langle 1,0\rangle$, and $\langle 0,0\rangle$ are the rekeying nodes, so that the key associated with $\langle 2,0\rangle$ is destructed and other keys are altered to new ones. In general, when a member $\mathrm{M}_{\langle l,m\rangle}$ joins a group, rekeying nodes are denoted as $\langle l-i, \lfloor \frac{m}{2^i} \rfloor \rangle$, where $i = 1, \ldots, l$. Similarly, when $\mathrm{M}_{\langle l,m\rangle}$ leaves a group, rekeying nodes (excluding a destructed node) are expressed as $\langle l-j, \lfloor \frac{m}{2^j} \rfloor \rangle$, where $j = 2, \ldots, l$.

3.5 Captain Selection

In a subtree that is rooted at a rekeying node, one of members of the subtree is selected as a *captain*, a representative member, to perform a DH key agreement and key distribution. In case of $\mathrm{M}_{\langle l,m\rangle}$ joining or leaving a group, there are l rekeying nodes; therefore l members who severally belong to the subtrees rooted at the rekeying nodes become their *captains*. From the view of a root node of a subtree, a *captain* of the subtree is selected from under the branch that does not lead to $\mathrm{M}_{\langle l,m\rangle}$, the opposite branch toward $\mathrm{M}_{\langle l,m\rangle}$. In the case of Fig. 2, the members $\mathrm{M}_{\langle 3,0\rangle}$, $\mathrm{M}_{\langle 3,2\rangle}$, and $\mathrm{M}_{\langle 3,4\rangle}$ become the *captains* $\mathrm{C}_{\langle 1,0\rangle}$, $\mathrm{C}_{\langle 2,1\rangle}$, and $\mathrm{C}_{\langle 0,0\rangle}$, respectively. In the Fig. 3, the members $\mathrm{M}_{\langle 3,1\rangle}$, $\mathrm{M}_{\langle 3,3\rangle}$, and $\mathrm{M}_{\langle 2,3\rangle}$ become the *captains* $\mathrm{C}_{\langle 2,0\rangle}$, $\mathrm{C}_{\langle 1,0\rangle}$, and $\mathrm{C}_{\langle 0,0\rangle}$.

The *captain* selection is performed by a deterministic algorithm, which is shared among all members of a group, not by leader election algorithms expecting strongly connected network[17]. When there is a join or a leave, each member independently selects a *captain* by the algorithm. There are two directions for constructing such algorithm: 1) The captaincy is taken over among all members of a subtree in turn; 2) Some criteria, such as computational power and bandwidth of each member, are used to decide a *captain*. The former is straightforward to make the load of rekeying flat, but the latter is effective when some members are powerful in computation compared to other members.

3.6 DH Key Agreement

A *captain* performs the DH key agreement to create new keys to be associated with the rekeying nodes. There are two strategies for the key agreement:

1. Dedicated Strategy: A single entity takes on the initiator roles of the protocol
2. Distributed Strategy: Several entities share the initiator roles of the protocol

The first strategy dedicates the initiator roles for the key agreements to a newcomer in the join protocol and to a *captain* who represents a subtree rooted

at the parent node of a seceder in the leave protocol. This strategy imposes relatively high computations on a single entity. The second strategy allots tasks of the key agreements on a newcomer and *captains* in the join protocol, and also tasks on *captains* in the leave protocol. This strategy is more effective to flatten the computations for the key agreements than the first one; however, it needs additional procedures in the key distribution (Section 3.7). In the following of this section, we explain our key agreement protocols along with the examples in Fig. 2 and Fig. 3. It is important to note that, the following protocols do not include any authentication process. To prevent the man-in-the-middle attack, some authentication process should be combined into the protocols.

DH Protocol on Dedicated Strategy *Join*: In Fig. 2, the newcomer $\mathrm{M}_{\langle 3,3\rangle}$ and the *captains* $\mathrm{M}_{\langle 3,2\rangle}$, $\mathrm{M}_{\langle 3,0\rangle}$, $\mathrm{M}_{\langle 3,4\rangle}$ compute the following expression using public parameters p and g, and their random secret $X_{\mathrm{M}_{\langle l,m\rangle}} \in Z_{p-1}$:

$$Y_{\mathrm{M}_{\langle l,m\rangle}} = g^{X_{\mathrm{M}_{\langle l,m\rangle}}} \bmod p \tag{1}$$

$\mathrm{M}_{\langle 3,3\rangle}$ sends a resultant value $Y_{\mathrm{M}_{\langle 3,3\rangle}}$ to the *captains* via unicast or multicast. The *captains* send resultant values to $\mathrm{M}_{\langle 3,3\rangle}$ via unicast or multicast as follows:

$$\begin{aligned}
&\mathrm{M}_{\langle 3,3\rangle} \rightarrow \{\mathrm{M}_{\langle 3,2\rangle}, \mathrm{M}_{\langle 3,0\rangle}, \mathrm{M}_{\langle 3,4\rangle}\}: \quad Y_{\mathrm{M}_{\langle 3,3\rangle}} \\
&\mathrm{M}_{\langle 3,2\rangle} \rightarrow \mathrm{M}_{\langle 3,3\rangle}: \quad Y_{\mathrm{M}_{\langle 3,2\rangle}} \\
&\mathrm{M}_{\langle 3,0\rangle} \rightarrow \mathrm{M}_{\langle 3,3\rangle}: \quad Y_{\mathrm{M}_{\langle 3,0\rangle}} \\
&\mathrm{M}_{\langle 3,4\rangle} \rightarrow \mathrm{M}_{\langle 3,3\rangle}: \quad Y_{\mathrm{M}_{\langle 3,4\rangle}}
\end{aligned}$$

Here the one value $Y_{\mathrm{M}_{\langle 3,3\rangle}}$ is used for three agreement procedures. If using the conventional DH protocol, the newcomer computes the expression (1) three times and sends three resultant values to the *captains* severally. Owing to this omission, loads of the newcomer can be reduced. As a result, $\mathrm{M}_{\langle 3,3\rangle}$ shares the following new keys with $\mathrm{M}_{\langle 3,2\rangle}$, $\mathrm{M}_{\langle 3,0\rangle}$, and $\mathrm{M}_{\langle 3,4\rangle}$, respectively, where $h()$ is a cryptographically secure hash function for the key derivation:

$$\begin{aligned}
\mathrm{K'}_{\langle 2,1\rangle} &= h(g^{X_{\mathrm{M}_{\langle 3,3\rangle}} X_{\mathrm{M}_{\langle 3,2\rangle}}} \bmod p) \\
\mathrm{K'}_{\langle 1,0\rangle} &= h(g^{X_{\mathrm{M}_{\langle 3,3\rangle}} X_{\mathrm{M}_{\langle 3,0\rangle}}} \bmod p) \\
\mathrm{K'}_{\langle 0,0\rangle} &= h(g^{X_{\mathrm{M}_{\langle 3,3\rangle}} X_{\mathrm{M}_{\langle 3,4\rangle}}} \bmod p)
\end{aligned}$$

We denote the above key agreement procedures as follows:

$$\begin{aligned}
&\mathrm{M}_{\langle 3,3\rangle} \leftrightarrow \mathrm{M}_{\langle 3,2\rangle}: \ \mathrm{K'}_{\langle 2,1\rangle} \\
&\mathrm{M}_{\langle 3,3\rangle} \leftrightarrow \mathrm{M}_{\langle 3,0\rangle}: \ \mathrm{K'}_{\langle 1,0\rangle} \\
&\mathrm{M}_{\langle 3,3\rangle} \leftrightarrow \mathrm{M}_{\langle 3,4\rangle}: \ \mathrm{K'}_{\langle 0,0\rangle}
\end{aligned}$$

***Leave*:** In Fig. 3, the *captains* compute the equation (1), and then the *captain* $\mathrm{M}_{\langle 3,1\rangle}$ sends $Y_{\mathrm{M}_{\langle 3,1\rangle}}$ to the other *captains*. The other *captains* send each resultant value to $\mathrm{M}_{\langle 3,1\rangle}$ as follows:

$$\begin{aligned}
&\mathrm{M}_{\langle 3,1\rangle} \rightarrow \{\mathrm{M}_{\langle 3,3\rangle}, \mathrm{M}_{\langle 2,3\rangle}\}: \quad Y_{\mathrm{M}_{\langle 3,1\rangle}} \\
&\mathrm{M}_{\langle 3,3\rangle} \rightarrow \mathrm{M}_{\langle 3,1\rangle}: \quad Y_{\mathrm{M}_{\langle 3,3\rangle}} \\
&\mathrm{M}_{\langle 2,3\rangle} \rightarrow \mathrm{M}_{\langle 3,1\rangle}: \quad Y_{\mathrm{M}_{\langle 2,3\rangle}}
\end{aligned}$$

$M_{\langle 3,1\rangle}$ shares following new keys between $M_{\langle 3,3\rangle}$ and $M_{\langle 2,3\rangle}$, respectively:

$$K'_{\langle 1,0\rangle} = h(g^{X_{M_{\langle 3,1\rangle}} X_{M_{\langle 3,3\rangle}}} \bmod p)$$
$$K'_{\langle 0,0\rangle} = h(g^{X_{M_{\langle 3,1\rangle}} X_{M_{\langle 2,3\rangle}}} \bmod p)$$

We denote the above key agreement procedures as follows:

$$M_{\langle 3,1\rangle} \leftrightarrow M_{\langle 3,3\rangle} : K'_{\langle 1,0\rangle}$$
$$M_{\langle 3,1\rangle} \leftrightarrow M_{\langle 2,3\rangle} : K'_{\langle 0,0\rangle}$$

DH Protocol on Distributed Strategy *Join*: In Fig. 2, the newcomer $M_{\langle 3,3\rangle}$ and the *captain* $M_{\langle 3,2\rangle}$ perform the DH key agreement protocol with the key derivation. $M_{\langle 3,2\rangle}$ also performs it with $M_{\langle 3,0\rangle}$. Lastly, $M_{\langle 3,0\rangle}$ performs it with $M_{\langle 3,4\rangle}$ as follows:

$$M_{\langle 3,3\rangle} \leftrightarrow M_{\langle 3,2\rangle} : K'_{\langle 2,1\rangle}$$
$$M_{\langle 3,2\rangle} \leftrightarrow M_{\langle 3,0\rangle} : K'_{\langle 1,0\rangle}$$
$$M_{\langle 3,0\rangle} \leftrightarrow M_{\langle 3,4\rangle} : K'_{\langle 0,0\rangle}$$

***Leave*:** In Fig. 3, the *captains* $M_{\langle 3,1\rangle}$ and $M_{\langle 3,3\rangle}$ perform the DH key agreement protocol with the key derivation. $M_{\langle 3,3\rangle}$ also performs it with $M_{\langle 2,3\rangle}$ as follows:

$$M_{\langle 3,1\rangle} \leftrightarrow M_{\langle 3,3\rangle} : K'_{\langle 1,0\rangle}$$
$$M_{\langle 3,3\rangle} \leftrightarrow M_{\langle 2,3\rangle} : K'_{\langle 0,0\rangle}$$

Note that a *captain* who performs the DH protocol twice, such as $M_{\langle 3,2\rangle}$ and $M_{\langle 3,0\rangle}$ in Fig. 2 and $M_{\langle 3,3\rangle}$ in Fig. 3, can merge the two values to be sent into one value using the same idea of the dedicated strategy.

3.7 Key Distribution

A *captain* distributes a new key that is derived from a DH key agreement to the subtree that it belongs to. A new key to be associated with a rekeying node is encrypted with the node's old key in a join protocol and is encrypted by a key associated with a child node of the rekeying node in a leave protocol. The encryptions on the FDLKH, however, are performed by *captains* not by a central server. Our key distribution scheme has slightly different two types of strategies, dedicated and distributed strategy. In the following, we explain our key distribution protocols along with the examples in Fig. 2 and Fig. 3.

Key Distribution on Dedicated Strategy *Join*: In Fig. 2, the *captains* $M_{\langle 3,0\rangle}$ and $M_{\langle 3,4\rangle}$ encrypt the new keys $K'_{\langle 1,0\rangle}$ and $K'_{\langle 0,0\rangle}$ by using the old keys $K_{\langle 1,0\rangle}$ and $K_{\langle 0,0\rangle}$ respectively and then send the encryptions to the subtrees $T_{\langle 1,0\rangle}$ and $T_{\langle 0,0\rangle}$ as follows:

$$M_{\langle 3,0\rangle} \rightarrow T_{\langle 1,0\rangle} : \quad E(K_{\langle 1,0\rangle}, K'_{\langle 1,0\rangle})$$
$$M_{\langle 3,4\rangle} \rightarrow T_{\langle 0,0\rangle} : \quad E(K_{\langle 0,0\rangle}, K'_{\langle 0,0\rangle})$$

***Leave*:** In Fig. 3, the *captains* $M_{\langle 3,3\rangle}$ and $M_{\langle 2,3\rangle}$ encrypt the new keys $K'_{\langle 1,0\rangle}$ and $K'_{\langle 0,0\rangle}$ by using the keys $K_{\langle 2,1\rangle}$ and $K_{\langle 1,1\rangle}$ respectively and then send the encryptions to the subtrees $T_{\langle 2,1\rangle}$ and $T_{\langle 1,1\rangle}$ as follows:

$$\begin{aligned} M_{\langle 3,3\rangle} \to T_{\langle 2,1\rangle} : &\quad E(K_{\langle 2,1\rangle}, K'_{\langle 1,0\rangle}) \\ M_{\langle 2,3\rangle} \to T_{\langle 1,1\rangle} : &\quad E(K_{\langle 1,1\rangle}, K'_{\langle 0,0\rangle}) \end{aligned}$$

After that, $M_{\langle 3,1\rangle}$ who knows all new keys encrypts the new key $K'_{\langle 0,0\rangle}$ by using the new key $K'_{\langle 1,0\rangle}$ that was sent on the previous step and performs a supplementary sending to the subtree $T_{\langle 1,0\rangle}$ as follows:

$$M_{\langle 3,1\rangle} \to T_{\langle 1,0\rangle} : \quad E(K'_{\langle 1,0\rangle}, K'_{\langle 0,0\rangle})$$

Key Distribution on Distributed Strategy *Join*: In Fig. 2, the *captains* $M_{\langle 3,0\rangle}$ and $M_{\langle 3,4\rangle}$ encrypt the new keys $K'_{\langle 1,0\rangle}$ and $K'_{\langle 0,0\rangle}$ by using the old keys $K_{\langle 1,0\rangle}$ and $K_{\langle 0,0\rangle}$ respectively and send the encryptions to the subtrees $T_{\langle 1,0\rangle}$ and $T_{\langle 0,0\rangle}$ as follows:

$$\begin{aligned} M_{\langle 3,0\rangle} \to T_{\langle 1,0\rangle} : &\quad E(K_{\langle 1,0\rangle}, K'_{\langle 1,0\rangle}) \\ M_{\langle 3,4\rangle} \to T_{\langle 0,0\rangle} : &\quad E(K_{\langle 0,0\rangle}, K'_{\langle 0,0\rangle}) \end{aligned}$$

At this moment, the newcomer $M_{\langle 3,3\rangle}$ only knows the new key $K'_{\langle 2,1\rangle}$; therefore, $M_{\langle 3,2\rangle}$ and $M_{\langle 3,0\rangle}$ encrypt the new keys $K'_{\langle 1,0\rangle}$ and $K'_{\langle 0,0\rangle}$ by using $K'_{\langle 2,1\rangle}$ and $K'_{\langle 1,0\rangle}$ respectively, and then send the encryptions to $M_{\langle 3,3\rangle}$ as follows:

$$\begin{aligned} M_{\langle 3,2\rangle} \to M_{\langle 3,3\rangle} : &\quad E(K'_{\langle 2,1\rangle}, K'_{\langle 1,0\rangle}) \\ M_{\langle 3,0\rangle} \to M_{\langle 3,3\rangle} : &\quad E(K'_{\langle 1,0\rangle}, K'_{\langle 0,0\rangle}) \end{aligned}$$

***Leave*:** In Fig. 3, the *captains* $M_{\langle 3,3\rangle}$ and $M_{\langle 2,3\rangle}$ encrypt the new keys $K'_{\langle 1,0\rangle}$ and $K'_{\langle 0,0\rangle}$ by using the keys $K_{\langle 2,1\rangle}$ and $K_{\langle 1,1\rangle}$ respectively and send the encryptions to the subtrees $T_{\langle 2,1\rangle}$ and $T_{\langle 1,1\rangle}$ as follows:

$$\begin{aligned} M_{\langle 3,3\rangle} \to T_{\langle 2,1\rangle} : &\quad E(K_{\langle 2,1\rangle}, K'_{\langle 1,0\rangle}) \\ M_{\langle 2,3\rangle} \to T_{\langle 1,1\rangle} : &\quad E(K_{\langle 1,1\rangle}, K'_{\langle 0,0\rangle}) \end{aligned}$$

After that, $M_{\langle 3,3\rangle}$ encrypts the new key $K'_{\langle 0,0\rangle}$ by using the new key $K'_{\langle 1,0\rangle}$ and performs a supplementary sending to the subtree $T_{\langle 1,0\rangle}$ as follows:

$$M_{\langle 3,3\rangle} \to T_{\langle 1,0\rangle} : \quad E(K'_{\langle 1,0\rangle}, K'_{\langle 0,0\rangle})$$

3.8 Key Update

After the key distributions, each member decrypts and obtains new keys. Finally each member associates the new keys with the rekeying nodes in one's own binary key tree.

The above mentioned join/leave protocols are generalized in the appendix B of this paper.

4 Evaluations

We evaluate the FDLKH from two points of view, number of keys and costs for a join and a leave. To make comparisons, we also apply the same criteria to the two centralized schemes, the basic and the LKH scheme mentioned in Section 2. For simplicity's sake, we assume that the degree of a tree for the LKH is 2 (i.e., a binary tree), the number of members before a leave or after a join is some power of 2, a tree forms a complete binary tree on the LKH and the FDLKH. In the situation, the equation $l = \log_2 n$ is valid when $\mathrm{M}_{\langle l,m \rangle}$ joins or leaves a group.

4.1 Number of Keys

Table 1 shows the total variety of keys in a group and the number of keys that a member of a group holds. The number of keys per member on the FDLKH is 1 less than that on the LKH because a member on the LKH has an individual key, whereas a member on the FDLKH does not have it. Owing to the individual keys, the total variety of keys in a group on the FDLKH is approximately half of that on the LKH.

Table 1. Number of keys

	Total	per Member
Basic	$n+1$	2
LKH	$2n-1$	$l+1$
FDLKH	$n-1$	l

4.2 Cost for a Join

Table 2 describes the costs for a join on the four schemes: basic, LKH, FDLKH on dedicated and distributed strategy. Here the costs mean the number of times of DH key agreement (DH), symmetric key encryption (Enc.) and decryption (Dec.) on each entity. In the table, *Key Server* means a central key management server; *Regular Member* indicates members of a group excepting *captains* and a newcomer. The dedicated strategy needs a newcomer to perform l times of the DH protocol[1], instead of l decryptions with an individual key on the LKH. I.e., the newcomer on the dedicated strategy is required relatively high computational power than the LKH[2]. In contrast, the distributed strategy needs a newcomer to perform the DH protocol only once and $l-1$ decryptions. On the LKH, the average number of decryptions on a member asymptotically comes close to

[1] As mentioned in Section 3.6, the FDLKH can omit a part of the DH protocol. Therefore, in fact, the computational cost of the FDLKH is less than the same times of the conventional two party DH protocol.

[2] Note that a newcomer on the LKH is required additional cost to share an individual key with the central server.

2[6]. The FDLKH inherits this property, therefore, a regular member performs approximately 2 decryptions (this property is also applied to the cost for a leave). The way to derive this average is shown in the appendix A of this paper. The load on the key server on the LKH, i.e., $2l$ encryptions, is distributed to the *captains* on the FDLKH. Fig. 4(a) and Fig. 4(b) illustrate the cost of symmetric key cryptosystem, which is the sum numbers of encryptions and decryptions, for a join on the dedicated/distributed strategy of the FDLKH. The cost on the regular member is derived from the equation (2) in the appendix A.

Table 2. Costs for a join

	Entity	DH	Enc.	Dec.
Basic	Newcomer	-	-	1
	per Member	-	-	1
	Key Server	-	2	-
LKH	Newcomer	-	-	l
	per Member	-	-	2
	Key Server	-	$2l$	-
FDLKH (dedicated)	Newcomer	l	-	-
	per Regular Member	-	-	2
	per Captain	1	1	$\frac{l-1}{2}$
FDLKH (distributed)	Newcomer	1	-	$l-1$
	per Regular Member	-	-	2
	per Captain	$\frac{2l-1}{l}$	$\frac{2l-2}{l}$	$\frac{(l-1)(l-2)}{2l}$

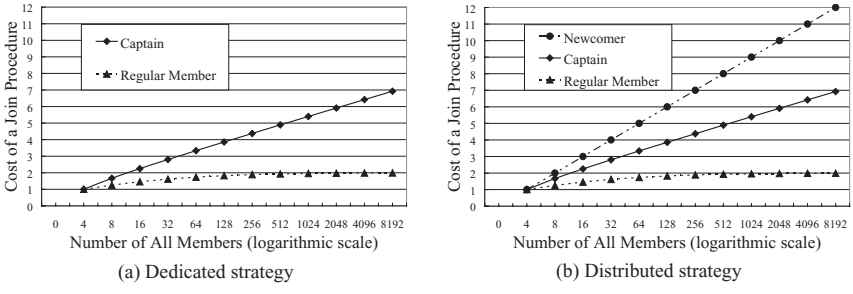

Fig. 4. Cost of symmetric key cryptosystem for a join on FDLKH

4.3 Cost for a Leave

Table 3 describes the costs of DH key agreement, symmetric key encryption and decryption for a leave on each scheme. In the table, *Regular Member* indicates members of a group excepting *captains* and a seceder; *Buddy Captain* means a

captain who represents a subtree rooted at the parent node of a seceder, that is, the neighboring *captain* of the seceder. The dedicated strategy of the FDLKH needs the *Buddy Captain* to conduct many tasks. In contrast, the distributed strategy allots the tasks to all *captains*. Fig. 5(a) and Fig. 5(b) illustrate the cost of symmetric key cryptosystem for a leave on the dedicated/distributed strategy of the FDLKH.

Table 3. Costs for a leave

	Entity	DH	Enc.	Dec.
Basic	Seceder	-	-	-
	per Member	-	-	1
	Key Server	-	$n-1$	-
LKH	Seceder	-	-	-
	per Member	-	-	2
	Key Server	-	$2l$	-
FDLKH (dedicated)	Seceder	-	-	-
	per Regural Member	-	-	2
	per Captain	1	1	$\frac{l-2}{2}$
	Buddy Captain	$l-1$	$l-2$	-
FDLKH (distributed)	Seceder	-	-	-
	per Regular Member	-	-	2
	per Captain	$\frac{2l-2}{l}$	$\frac{2l-3}{l}$	$\frac{(l-1)(l-2)}{2l}$

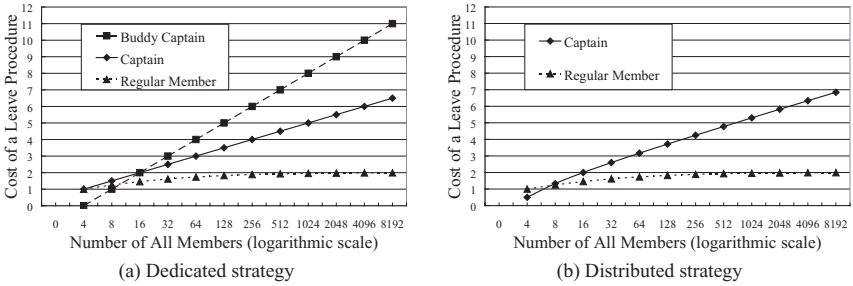

Fig. 5. Cost of symmetric key cryptosystem for a leave on FDLKH

5 Conclusion and Future Works

In this paper, we proposed a fully decentralized key management scheme that does not require any central server to update a group key. Our scheme FDLKH is established with the help of *captains* who are ones of members of a group and perform the key agreement and key distribution as the representatives of subgroups in the group, instead of a central server. The FDLKH has two kinds of

strategies, that is, the dedicated and distributed strategy. In the former strategy, a single entity dedicates its computational power to the rekeying. In the latter strategy, the computations for the rekeying are further distributed. Whether in the dedicated strategy or in the distributed strategy, each member of the group only knows the keys that it ought to know. We estimated the number of keys on the FDLKH. Since the FDLKH does not need individual keys, the number of keys per member was 1 less than that of the LKH. This reduction cut down the total variety of keys on the FDLKH to approximately half of that of the LKH. We also estimated the cost for a join and leave. Because the FDLKH inherits the efficiency of the rekeying mechanism on the LKH, the cost was kept $O(\log n)$, furthermore, it was highly distributed among the members in the group. The algorithm of the *captain* selection is one of the important components of the FDLKH. We described a couple of directions for designing the algorithm, that is, taking over the captaincy in turn and deciding the *captains* by some criteria (e.g., computational power). We will further discuss detailed algorithms for the *captain* selection that does not need a central server.

References

1. Yoshida, M., Kuroda, M., Kiyomoto, S., Tanaka, T.: A secure service architecture for beyond 3G wireless network. Proc. 6th International Symposium on Wireless Personal Multimedia Communications, Vol. 2 (2003) 579-583
2. Kuroda, M., Inoue, M., Okubo, A., Sakakura, T., Shimizu, K., Adachi, F.: Scalable mobile ethernet and fast vertical handover. Proc. IEEE Wireless Communications and Networking Conference 2004 (2004) A27-3
3. Harney, H., Muckenhirn, C., Rivers, T.: Group key management protocol (GKMP) specification. IETF, RFC 2093 (1997)
4. Harney, H., Muckenhirn, C., Rivers, T.: Group key management protocol (GKMP) architecture. IETF, RFC 2094 (1997)
5. Wallner, D., Harder, E., Agee, R.: Key management for multicast: issues and architectures. IETF, RFC 2627 (1999)
6. Wong, C. K., Gouda, M., Lam, S.: Secure group communication using key graphs. IEEE/ACM Trans. on Networking, Vol. 8, No. 1 (2000) 16-30
7. Canetti, R., Garay, J., Itkis, G., Micciancio, D., Naor, M., Pinkas, B.: Multicast security: a taxonomy and efficient constructions. Proc. IEEE Infocom '99, Vol.2 (1999) 708-716
8. McGrew, D.A., Sherman, A.T.: Key establishment in large dynamic groups using one-way function trees. IEEE Trans. on Software Engineering, Vol. 29, No. 5 (2003) 444-458
9. Perrig, A., Song, D., Tygar, J.D.: ELK: a new protocol for efficient large-group key distribution. Proc. 2001 IEEE Security and Privacy Symposium (2001) 247-262
10. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J. D.: SPINS: security protocols for sensor networks. Proc. Mobile Computing and Networking 2001 (2001) 189-199
11. Law, Y. W., Corin, R., Etalle, S., Hartel, P. H.: A formally verified decentralized key management architecture for wireless sensor networks. Proc. Personal Wireless Communications 2003 (2003) 27-39

12. Steer, D. G., Strawczynski, L., Diffie, W., Wiener, M.: A secure audio teleconference system. Proc. Advances in Cryptology-CRYPTO '88 (1988) 520-528
13. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system. Proc. Advances in Cryptology-EUROCRYPT '94 (1994) 275-286
14. Steiner, M., Tsudik, G., Waidner, M.: Key agreement in dynamic peer groups. IEEE Trans. on Parallel and Distributed Systems, Vol. 11, No. 8 (2000) 769-780
15. Alves-Foss, J.: An efficient secure authenticated group key exchange algorithm for large and dynamic groups. Proc. 23rd National Information Systems Security Conference (2000) 254-266
16. Kim, Y., Perrig, A., Tsudik, G.: Simple and fault-tolerant key agreement for dynamic collaborative groups. ACM Conference on Computer and Communications Security 2000 (2000) 235-244
17. Lynch, N.: Distributed algorithms. Morgan Kaufmann Publishers (1996)

Appendix A: Number of Decryptions on a Regular Member

Let n be the number of all members in a group, l be the number of *captains*. Since there is a newcomer or a seceder, the number of regular members is $n - l - 1$. Here we assume that n is some power of 2 and a key tree forms a complete binary tree. The regular members in half of the tree, $\frac{n}{2} - 1$ members, perform one decryption; the regular members in further half of the remaining tree, $\frac{n}{2^2} - 1$ members, perform two decryptions. Continuing these calculations, the average number of decryptions on a regular member derives as follows:

$$\frac{\sum_{t=1}^{l-1}(\frac{n}{2^t} - 1)t}{n - l - 1} = \frac{2 - (\frac{l+1}{2^{l-1}} + \frac{l(l-1)}{2})}{1 - \frac{l+1}{n}} \tag{2}$$

$$< \frac{2}{1 - \frac{l+1}{n}} \tag{3}$$

The expression (3) asymptotically comes close to 2 according to increasing n.

Appendix B: Generalized Protocols

In this appendix, we generalize the join and leave protocols of the FDLKH on both the dedicated and distributed strategies. It is assumed that $\mathrm{M}_{\langle l,m \rangle}$ is the newcomer/seceder who passed some authentication process of a group and sent a join/leave request to the group. Procedures including integers *i, j, k* are the repetitive procedures that are repeated according to the integers. Some key distribution procedures marked '$*$' are performed by a *captain* only when the target subtree, i.e., destination of the key distribution, has one or more members excluding the *captain*.

Join Protocol on the Dedicated Strategy

(1) **if** $(l = 1)$
(2) **Update** $\mathrm{T}_{\langle 0,0\rangle}$;
(3) **Select** $\mathrm{C}_{\langle 0,0\rangle}$;
(4) $\mathrm{M}_{\langle 1,1\rangle} \leftrightarrow \mathrm{C}_{\langle 0,0\rangle} : \mathrm{K}'_{\langle 0,0\rangle}$;
(5)∗ $\mathrm{C}_{\langle 0,0\rangle} \rightarrow \mathrm{T}_{\langle 1,0\rangle} : \mathrm{E}(\mathrm{K}_{\langle 1,0\rangle}, \mathrm{K}'_{\langle 0,0\rangle})$;
(6) $\mathrm{K}_{\langle 0,0\rangle} := \mathrm{K}'_{\langle 0,0\rangle}$;

(7) **else if** $(l \geq 2)$
(8) **Update** $\mathrm{T}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle}$;
(9) **Select** $\mathrm{C}_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle}$;
(10) $\mathrm{M}_{\langle l,m\rangle} \leftrightarrow \mathrm{C}_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle} : \mathrm{K}'_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle}$;
(11) $\mathrm{C}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle}, \mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle})$;
(12)∗ $\mathrm{C}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l,m+(-1)^m\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle l,m+(-1)^m\rangle}, \mathrm{K}'_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle})$;
(13) $\mathrm{K}_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle} := \mathrm{K}'_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle}$;

∗ if needed

Join Protocol on the Distributed Strategy

(1) **if** $(l = 1)$
(2)-(6) Similar to the join protocol on the dedicated strategy

(7) **else if** $(l \geq 2)$
(8) **Update** $\mathrm{T}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle}$;
(9) **Select** $\mathrm{C}_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle}$;
(10) $\mathrm{M}_{\langle l,m\rangle} \leftrightarrow \mathrm{C}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle} : \mathrm{K}'_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle}$;
(11) $\mathrm{C}_{\langle l-(j-1),\lfloor \frac{m}{2^{j-1}} \rfloor\rangle} \leftrightarrow \mathrm{C}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} :$
$\mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle}$;
(12) $\mathrm{C}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle}, \mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle})$;
(13)∗ $\mathrm{C}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l,m+(-1)^m\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle l,m+(-1)^m\rangle}, \mathrm{K}'_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle})$;
(14) $\mathrm{C}_{\langle l-(j-1),\lfloor \frac{m}{2^{j-1}} \rfloor\rangle} \rightarrow \mathrm{M}_{\langle l,m\rangle} :$
$\mathrm{E}(\mathrm{K}'_{\langle l-(j-1),\lfloor \frac{m}{2^{j-1}} \rfloor\rangle}, \mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle})$;
(15) $\mathrm{K}_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle} := \mathrm{K}'_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle}$;

∗ if needed

Leave Protocol on the Dedicated Strategy

(1) **if** $(l = 1)$
(2) **Remove** $\langle 0,0\rangle$;
(3) **Update** $\mathrm{T}_{\langle 1,m+(-1)^m\rangle}$;

(4) **else if** $(l = 2)$
(5) **Select** $\mathrm{C}_{\langle 1,\lfloor \frac{m}{2} \rfloor\rangle}$ and $\mathrm{C}_{\langle 0,0\rangle}$;
(6) $\mathrm{C}_{\langle 1,\lfloor \frac{m}{2} \rfloor\rangle} \leftrightarrow \mathrm{C}_{\langle 0,0\rangle} : \mathrm{K}'_{\langle 0,0\rangle}$;
(7)∗ $\mathrm{C}_{\langle 0,0\rangle} \rightarrow \mathrm{T}_{\langle 1,\lfloor \frac{m}{2} \rfloor+(-1)^{\lfloor \frac{m}{2} \rfloor}\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle 1,\lfloor \frac{m}{2} \rfloor+(-1)^{\lfloor \frac{m}{2} \rfloor}\rangle}, \mathrm{K}'_{\langle 0,0\rangle})$;
(8)∗ $\mathrm{C}_{\langle 1,\lfloor \frac{m}{2} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle 2,m+(-1)^m\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle 2,m+(-1)^m\rangle}, \mathrm{K}'_{\langle 0,0\rangle})$;
(9) $\mathrm{K}_{\langle 0,0\rangle} := \mathrm{K}'_{\langle 0,0\rangle}$;
(10) **Update** $\mathrm{T}_{\langle 1,\lfloor \frac{m}{2} \rfloor\rangle}$;

(11) **else if** $(l \geq 3)$
(12) **Select** $\mathrm{C}_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle}$;
(13) $\mathrm{C}_{\langle l-1,\lfloor \frac{m}{2^j} \rfloor\rangle} \leftrightarrow \mathrm{C}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} : \mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle}$;
(14)∗ $\mathrm{C}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l-(j-1),\lfloor \frac{m}{2^{j-1}} \rfloor+(-1)^{\lfloor \frac{m}{2^{j-1}} \rfloor}\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle l-(j-1),\lfloor \frac{m}{2^{j-1}} \rfloor+(-1)^{\lfloor \frac{m}{2^{j-1}} \rfloor}\rangle}, \mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle})$;
(15)∗ $\mathrm{C}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l,m+(-1)^m\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle l,m+(-1)^m\rangle}, \mathrm{K}'_{\langle l-2,\lfloor \frac{m}{2^2} \rfloor\rangle})$;
(16) $\mathrm{C}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l-k,\lfloor \frac{m}{2^k} \rfloor\rangle} :$
$\mathrm{E}(\mathrm{K}'_{\langle l-k,\lfloor \frac{m}{2^k} \rfloor\rangle}, \mathrm{K}'_{\langle l-(k+1),\lfloor \frac{m}{2^{k+1}} \rfloor\rangle})$;
(17) $\mathrm{K}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} := \mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle}$;
(18) **Update** $\mathrm{T}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle}$;

∗if needed

Leave Protocol on the Distributed Strategy

(1) **if** $(l = 1)$
(2)-(3) Similar to the leave protocol on the dedicated strategy

(4) **else if** $(l = 2)$
(5)-(10) Similar to the leave protocol on the dedicated strategy

(11) **else if** $(l \geq 3)$
(12) **Select** $\mathrm{C}_{\langle l-i,\lfloor \frac{m}{2^i} \rfloor\rangle}$;
(13) $\mathrm{C}_{\langle l-(j-1),\lfloor \frac{m}{2^{j-1}} \rfloor\rangle} \leftrightarrow \mathrm{C}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} : \mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle}$;
(14)∗ $\mathrm{C}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l-(j-1),\lfloor \frac{m}{2^{j-1}} \rfloor+(-1)^{\lfloor \frac{m}{2^{j-1}} \rfloor}\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle l-(j-1),\lfloor \frac{m}{2^{j-1}} \rfloor+(-1)^{\lfloor \frac{m}{2^{j-1}} \rfloor}\rangle}, \mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle})$;
(15)∗ $\mathrm{C}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l,m+(-1)^m\rangle} :$
$\mathrm{E}(\mathrm{K}_{\langle l,m+(-1)^m\rangle}, \mathrm{K}'_{\langle l-2,\lfloor \frac{m}{2^2} \rfloor\rangle})$;
(16) $\mathrm{C}_{\langle l-k,\lfloor \frac{m}{2^k} \rfloor\rangle} \rightarrow \mathrm{T}_{\langle l-k,\lfloor \frac{m}{2^k} \rfloor\rangle} :$
$\mathrm{E}(\mathrm{K}'_{\langle l-k,\lfloor \frac{m}{2^k} \rfloor\rangle}, \mathrm{K}'_{\langle l-(k+1),\lfloor \frac{m}{2^{k+1}} \rfloor\rangle})$;
(17) $\mathrm{K}_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle} := \mathrm{K}'_{\langle l-j,\lfloor \frac{m}{2^j} \rfloor\rangle}$;
(18) **Update** $\mathrm{T}_{\langle l-1,\lfloor \frac{m}{2} \rfloor\rangle}$;

∗if needed